Modelos de computación lógica con ADN

Autor: Iñaki Sainz de Murieta Fuentes

Tutor: Alfonso Rodríguez-Patón Aradas

Universidad Politécnica de Madrid - Facultad de Informática

Tesis presentada para la obtención del título de

Máster de Investigación en Inteligencia Artificial

Septiembre 2011

Resumen

La computación molecular es una disciplina que se ocupa del diseño e implementación de dispositivos para el procesamiento de información sobre un sustrato biológico, como el ácido desoxirribonucleico (ADN), el ácido ribonucleico (ARN) o las proteínas. Desde que Watson y Crick descubrieron en los años cincuenta la estructura molecular del ADN en forma de doble hélice, se desencadenaron otros descubrimientos como las enzimas que cortan el ADN o la reacción en cadena de la polimerasa (PCR), contribuyendo más que significativamente a la irrupción de la *tecnología del ADN recombinante*.

Gracias a esta tecnología y al descenso vertiginoso de los precios de secuenciación y síntesis del ADN, la computación biomolecular pudo abandonar su concepción puramente teórica. En 1994, Leonard Adleman logró resolver un problema de computación NP-completo (*El Problema del Camino de Hamilton Dirigido*) utilizando únicamente moléculas de ADN. La gran capacidad de procesamiento en paralelo ofrecida por las técnicas del ADN recombinante permitió a Adleman ser capaz de resolver dicho problema en tiempo polinómico, aunque a costa de un consumo exponencial de moléculas de ADN. Utilizando algoritmos similares al de "fuerza bruta" utilizado por Adleman se logró resolver otros problemas NP-completos (por ejemplo, el de *Satisfacibilidad de Fórmulas Lógicas / SAT*).

Pronto se comprendió que la computación con biomolecular no podía competir en velocidad ni precisión con los ordenadores de silicio, por lo que su enfoque y objetivos se centraron en la resolución de problemas biológicos con aplicación biomédica, dejando de lado la resolución de problemas clásicos de computación. Desde entonces se han propuesto diversos modelos de dispositivos biomoleculares que, de forma autónoma (sin necesidad de un bio-ingeniero realizando operaciones de laboratorio), son capaces de procesar

como entrada un sustrato biológico y proporcionar una salida también en formato biológico: procesadores que aprovechan la extensión de la Polimerasa, autómatas que funcionan con enzimas de restricción o con deoxiribozimas, circuitos de hibridación competitiva.

Esta tesis presenta un conjunto de modelos de dispositivos de ácidos nucleicos escalables, sensibles al tiempo y energéticamente eficientes, capaces de implementar diversas operaciones de computación lógica aprovechando el fenómeno de la *hibridación competitiva del ADN*. La capacidad implícita de estos dispositivos para aplicar reglas de inferencia como *modus ponens*, *modus tollens*, *resolución* o el *silogismo hipotético* tiene un gran potencial. Entre otras funciones, permiten representar implicaciones lógicas (o reglas del tipo SI/ENTONCES), como por ejemplo, "si se da el síntoma 1 y el síntoma 2, entonces estamos ante la enfermedad A", o "si estamos ante la enfermedad B, entonces deben manifestarse los síntomas 2 y 3". Utilizando estos módulos lógicos como bloques básicos de construcción, se pretende desarrollar sistemas in vitro basados en sensores de ADN, capaces de trabajar de manera conjunta para detectar un conjunto de síntomas de entrada y producir una diagnóstico de salida. La reciente publicación en la revista *Science* de un autómata biomolecular de diagnóstico, capaz de tratar las células cancerígenas sin afectar a las células sanas, es un buen ejemplo de la relevancia científica que este tipo de autómatas tienen en la actualidad.

Además de las recién mencionadas aplicaciones en el diagnóstico *in vitro*, los modelos presentados también tienen utilidad en el diseño de biosensores inteligentes y la construcción de bases de datos con registros en formato biomolecular que faciliten el análisis genómico.

El estudio sobre el estado de la cuestión en computación biomolecular que se presenta en esta tesis está basado en un artículo recientemente publicado en la revista *Current Bioinformatics*. Los nuevos dispositivos presentados en la tesis forman parte de una solicitud de patente de la que la UPM es titular, y han sido presentados en congresos internacionales como *Unconventional Computation 2010* en Tokio o *Synthetic Biology 2010* en París.

Dedicatoria

A mi novia María Noel, por su paciencia con mis estudios, por estar siempre a mi lado y hacer de este "friki" un ser más sensible y humano.

A mi familia, por apoyarme en todos los desafíos que me he planteado en la vida. Gracias a ellos siempre será posible una próxima dedicatoria.

A Alfonso, por su tiempo, su apoyo, su guía y su consejo. Y por ser capaz de contagiarme su pasión por la ciencia.

Índice general

Índice de Figuras

ÍNDICE DE FIGURAS

Capítulo 1

Introducción

1.1. Computación biomolecular y la tecnología del ADN recombinante

La *computación biomolecular* (también denominada *biocomputación*) es una disciplina científica que se ocupa del diseño e implementación de dispositivos para el procesamiento de información sobre un sustrato de macromoléculas biológicas, como lo son el ácido desoxirribonucleico (ADN), el ácido ribonucleico (ARN) o las proteínas (1). Sin duda, es la molécula de ADN la que ha tomado más protagonismo en su empleo como sustrato para este paradigma de computación, que frecuentemente es también denominado *computación con ADN*.

La elección del ADN como sustrato biológico sobre el que realizar procesos de computación no ha sido fruto de la casualidad.

- Desde que en 1953 Watson y Crick descubrieron la estructura molecular del ADN en forma de doble hélice (2), el ADN ha sido el centro de estudio de la bioquímica y la genética. Prosiguieron descubrimientos como el de las *enzimas de restricción* (enzimas que cortan el ADN) (3, 4) en los años setenta o la *reacción en cadena de la Polimerasa* (PCR) (5) en los años ochenta, contribuyendo muy significativamente a la irrupción de la denominada *tecnología del ADN recombinante*. Esta tecnología abarca los métodos y procedimientos de laboratorio que permiten manipular y utilizar el ADN para realizar sobre él distintos tipos de operaciones.

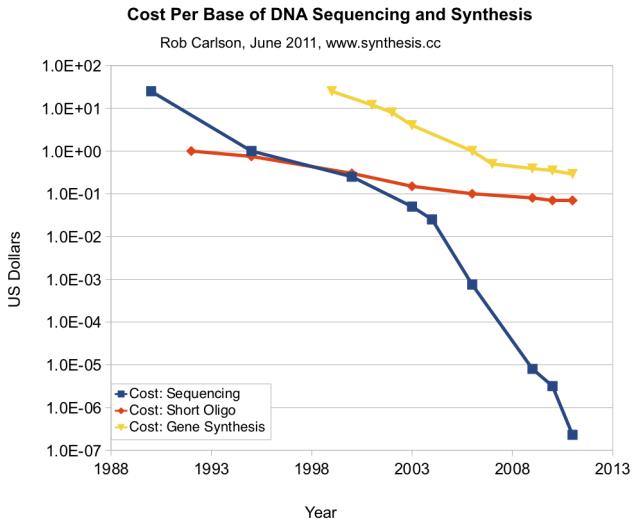

Figura 1.1: En los últimos veinte años, tanto el precio de sintetización como el se secuenciación de ADN han descendido a ritmo exponencial. Gráfica realizada por Robert Carlson, http://www.synthesis.cc/assets_c/2011/06/carlson_cost%20per_base_june_2011.html.

- Desde los años noventa, la tecnología del ADN recombinante se ha desarrollado a un ritmo vertiginoso. Una clara muestra de ello son los precios de secuenciación y síntesis del ADN, que desde el comienzo de dicha década, mostrando un curioso paralelismo con el silicio y la *Ley de Moore*, no han parado de caer a ritmo exponencial (ver figura 1.1).

Gracias a la irrupción de la tecnología del ADN recombinante, la computación biomolecular, que hasta entonces había permanecido en un plano puramente teórico, pudo demostrar su viabilidad de implementación. En 1994, Leonard Adleman logró resolver en 1994 un problema de computación complejo utilizando moléculas de ADN (6): el *Problema del Camino de Hamilton Dirigido*, consistente en encontrar un camino que visite todos los vértices de un grafo sólamente una vez. Dicho problema pertenece a la clase de complejidad NP-completo, lo que significa que su resolución es extremadamente compleja, no habiendo ningún algoritmo conocido que lo resuelva en tiempo polinómico. Hasta ese momento, nadie había sido capaz de resolver ese tipo de problemas complejos utilizando biomoléculas como medio de computación y aprovechando la gran capacidad de procesamiento en paralelo ofrecida por las técnicas del ADN recombinante. Este trabajo supuso el comienzo de la computación biomolecular con ADN.

Las extraordinarias capacidades del ADN sacadas a la luz por los trabajos de Adleman, tanto para almacenar como para procesar información en paralelo, crearon grandes expectativas en la comunidad científica. Considerando que un tubo de ensayo puede contener alrededor de 10^{20} hebras de ADN, que pueden ser empleadas en codificar todas las soluciones de un problema, y que las operaciones para todas las hebras del tubo pueden realizarse en paralelo, la expectación estaba más que justificada. Otros logros similares se sucedieron poco tiempo después, resolviendo más problemas NP-completos utilizando ADN (7, 8, 9). Pero la euforia no duró mucho tiempo: pronto se fue tomando conciencia de las limitaciones de este nuevo modelo computacional.

En primer lugar, se comprendió que los problemas NP-completos seguían siendo intratables cuando se usaba un algoritmo basado en fuerza bruta como el de Adleman. Con su solución, el tiempo dejaba de ser el recurso consumido exponencialmente, pero siempre a costa de un consumo exponencial de espacio: el número de moléculas necesarias para codificar todas las potenciales soluciones crecía de manera exponencial con el tamaño del problema. Además, las operaciones de laboratorio implicadas en este tipo de bioalgoritmos están sujetas a diversas fuentes de error, y elegir las operaciones que minimizan la tasa de error no es trivial. Pronto se comprendió que los procesadores moleculares no iban a poder competir en velocidad y precisión con los ordenadores de silicio.

Así pues, el enfoque y los objetivos de la computación con ADN tuvo que cambiar después de esos años, dejando de lado la resolución de problemas clásicos de computación para centrarse en problemas biológicos de aplicación biomédica. Es aquí donde los procesadores de ADN encuentran su aplicación ideal, gracias a su capacidad natural para detectar y procesar información biológica. El autómata de Benenson (10) fue uno de los primeros trabajos en mostrar este cambio de enfoque de manera significativa.

Diversos modelos de procesadores de ADN han sido implementados desde este enfoque. Pueden encontrarse ejemplos de autómatas de ADN que funcionan con enzimas de restricción (10, 11, 12, 13), circuitos de hibridación competitiva (14, 15, 16, 17, 18, 19, 20, 21), autómatas basados en deoxiribozimas (22, 23, 24, 25, 26, 27), y procesadores

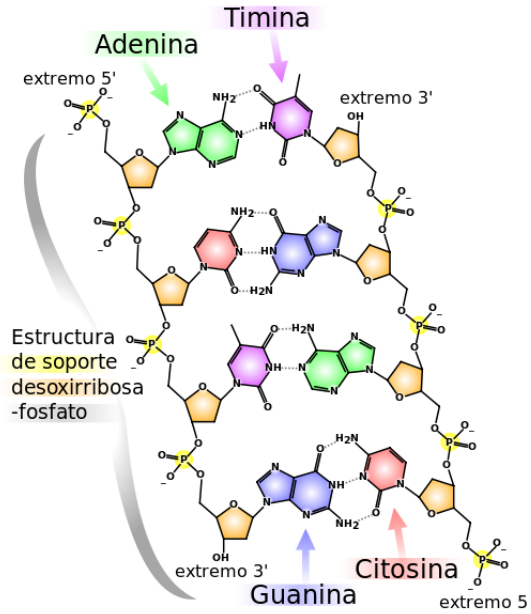

Figura 1.2: Estructura de la molécula de ADN. Obtenida bajo la licencia Creative Commons Genérica de Atribución/Compartir-Igual 2.5.

basados en ARN (28, 29, 30, 31). Todos ellos van a ser analizados en detalle en este capítulo.

1.2. El ADN y sus operaciones básicas

El *ácido desoxiribonucleico (ADN)* está presente en todas las células de los seres vivos. Normalmente se presenta como una gran molécula, compuesta por dos hebras que forman una estructura de doble hélice. Cada una de dichas hebras es un polímero largo formado por una secuencia de nucleótidos, cada uno de los cuales está compuesto por un grupo fosfato, un azúcar (desoxiribosa) y una base. La unión entre ambas hebras de ADN es posible gracias a la complementariedad de Watson-Crick (2), que establece la existencia de una fuerza de atracción entre bases complementarias: la adenina (A) se une siempre a la timina (T), mientras que la citosina (C) se une siempre a la guanina (G) (en el caso del ARN, la base uracilo (U) reemplaza a la timina). La figura 1.2 muestra en detalle la estructura de la molécula de ADN.

En *ácido ribonucleico (ARN)* tiene también una estructura muy similar, presentando las siguientes diferencias respecto al ADN:

- La base uracilo (U) reemplaza a la timina (T).
- En lugar de desoxiribosa, la molécula de azúcar empleada para unir la base al grupo fosfato es una ribosa.

Basta una simple observación de la figura 1.2 para comprobar que los ácidos nucleicos (tanto ADN como ARN) son moléculas asimétricas y direccionales. Los distintos nucleótidos están unidos entre sí gracias a enlaces fosfodiéster, que conectan el grupo hidroxilo 3' de la deoxiribosa (ribosa en el ARN) de un nucleótido con el grupo fosfato 5' del siguiente nucleótido. Como consecuencia de esta asimetría, las hebras de ácidos nucleicos tienen dos extremos diferenciados: el *extremo 3'*, caracterizado porque el nucleótido en ese extremo está unido a un grupo hidroxilo (OH), y el *extremo 5'*, caracterizado porque el nucleótido en ese extremo está unido a un grupo fosfato libre, no compartido con ningún otro nucleótido de la hebra. La direccionalidad de las moléculas de ADN y ARN viene implícita con su asimetría. Por convenio, las secuencias de bases de las hebras se escriben desde el extremo 5' al 3', ya que los procesos de transcripción[1], traducción[2] y replicación[3] ocurren en esa dirección. En consecuencia, cuando las hebras de ácidos nucleicos se representan con aristas, se dibuja una punta de flecha en el extremo 3' en analogía a los procesos biológicos mencionados.

Para que dos hebras de ácidos nucleicos hibriden entre sí, deben ser complementarias de forma antiparalela. Esto significa que, además de que cada base esté apareada con su pareja complementaria Watson-Crick, el extremo 3' de la primera hebra debe estar unido extremo 5' de la segunda hebra, y viceversa (ver figura 1.3). Esta característica va a ser de vital importancia a la hora de diseñar algoritmos y dispositivos con ácidos

[1] La transcripción del ADN es el primer paso de la expresión génica, mediante el cual se transfiere la información contenida en el ADN a una cadena de ARN mensajero. La enzima *ARN polimerasa* es la responsable del proceso.

[2] La tradución del ARN es el segundo paso de la expresión génica, responsable de sintetizar aminoácidos a partir de la información contenida en el ARN mensajero. Los *ribosomas* y el *ARN de transferencia* son los responsables del proceso.

[3] La replicación del ADN es el proceso que permite al ADN duplicarse, sintetizando así una copia idéntica. El *ADN polimerasa* es el responsable del proceso.

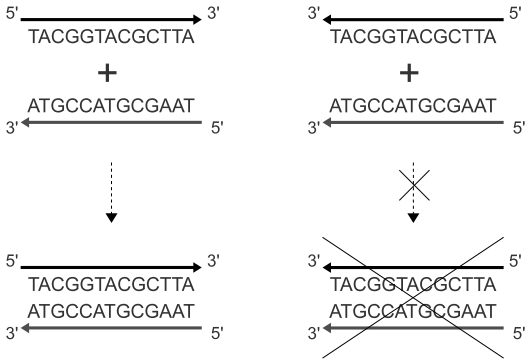

Figura 1.3: Hibridación antiparalela de las hebras de ácidos nucleicos.

nucleicos, para poder predecir y evaluar la estructura de las moléculas formadas.

Otra característica muy importante en la hibridación de hebras de ácidos nucleicos es la fuerza de sus enlaces entre bases. Si consideramos cada uno de estos enlaces de manera individual, la atracción que experimentan es débil. Para conseguir moléculas estables de ácidos nucleicos en hebra doble, es necesario que las secuencias de bases que las integran sean suficientemente largas. En cada momento, siempre hay una fracción de las bases que liberan sus enlaces temporalmente (ver figura 1.4). Este fenómeno se denomina *respiración de las bases* (del inglés, *base breathing*). Así pues, cuando más largas sean las hebras de las moléculas hibridadas, mayor probabilidad hay de estas que permanezcan unidas frente a la irrupción espontánea de los fenómenos de respiración.

Si nos aproximamos a la estructura de los ácidos nuleicos y sus propiedades desde el punto de vista de un informático, podemos afirmar que cada hebra de ADN codifica una cadena del alfabeto {A,G,C,T} y que cada hebra de ARN codifica una cadena del alfabeto {A,C,C,U}. Dada una cadena x codificada con el alfabeto {A,G,C,T}, la denotaremos como $\uparrow x$ si el sentido de las letras avanza hacia el extremo 3'. En cambio, $\downarrow x$ codifica la hebra complementaria a $\uparrow x$, con el sentido de las letras avanzando hacia el extremo 5'. Cuando $\uparrow x$ y $\downarrow x$ se aproximan, ambas hebras hibridan entre sí por complementariedad de bases, formando una doble hebra de ADN denotada con $\updownarrow x$. Normalmente, esta notación se simplifica usando x en lugar de $\uparrow x$ y $\bar{x}$ en lugar de $\downarrow x$.

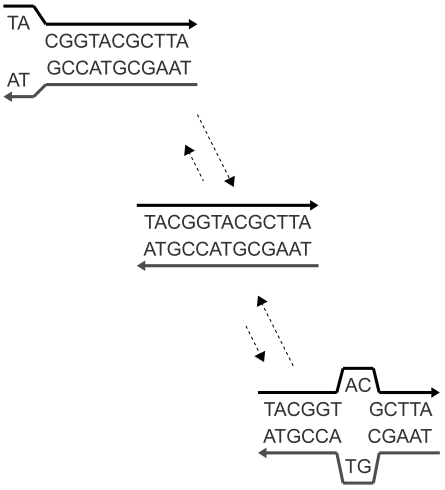

Figura 1.4: Respiración de las bases de las moléculas de ADN.

La computación con biomolecular utiliza las hebras de ADN como elemento de memoria que codifica la información necesaria para resolver un problema. Pero también es necesario un conjunto de operaciones que permitan procesar esas hebras en un laboratorio siguiendo un bioalgoritmo concreto. A continuación se detallan algunas de las más comúnmente utilizadas:

Síntesis Elaboración de moléculas de ADN con una secuencia específica.

Secuenciación Dada una hebra de ADN, determinar la secuencia de bases que la componen.

Hibridación Dadas dos hebras de ADN cuyas secuencias de bases son complementarias, al encontrarse bajo las condiciones de pH y temperatura adecuadas, hibridarán entre sí formando una molécula de ADN de hebra doble.

Desnaturalización Dada una hebra doble de ADN, un incremento de temperatura podrá lograr que ambas hebras se separen (o desnaturalicen).

Corte Las *enzimas de restricción* son una clase de proteína capaz de reconocer una secuencia de bases específica en una cadena doble de ADN y cortar ambas hebras de una manera precisa. Las zonas de reconocimiento y de corte son específicas de cada enzima, y no necesariamente coinciden.

Concatenación Existe una clase de enzima llamada *ligasa*, capaz de concatenar dos hebra de ADN uniendo el extremo 3' de la primera hebra con el extremo 5' de la segunda (o viceversa).

Filtrado por longitud Se realiza mediante la técnica de la *electroforesis en gel*, consistente en la aplicación de una corriente eléctrica a través de un gel poroso que contiene moléculas de ADN. Como dichas moléculas están cargadas negativamente, la corriente les hace moverse hacia el cátodo. Como las hebras más cortas se mueven más rápidamente a través de los poros, el proceso irá colocando las moléculas de ADN en diferentes ranuras de acuerdo a sus distintas longitudes de hebra (cuanto más pequeñas, más cerca del cátodo; cuanto más grandes, más cerca del ánodo). Tras el proceso de filtrado, el ADN de una longitud determinada puede extraerse de su ranura y ser reutilizado en otras operaciones.

Separación por afinidad Extracción de todas las hebras que contienen una secuencia específica de bases. Para extraer hebras que contengan la secuencia $\uparrow x$, se sintetizan hebras complementarias de secuencia $\downarrow x$ (llamadas *sondas*) y se fijan a un soporte sólido. Posteriormente se introduce el soporte con las sondas en el tubo de ensayo, que contiene ADN desnaturalizado. Al enfriar la solución, las hebras que contengan la secuencia $\uparrow x$ hibridarán con las sondas $\downarrow x$, quedando así las hebras objetivo fijadas al soporte sólido.

Polimerización con ADN Polimerasa Generación de la hebra de ADN complementaria a una hebra simple de ADN dada. Además de la enzima ADN Polimerasa y de la hebra original $\uparrow xyz$, es necesario añadir a la solución pequeñas hebras parcialmente complementarias a la original (llamadas *cebadores*), $\downarrow z$. Tras hibridarse $\uparrow xyz$ y $\downarrow z$, el ADN Polimerasa se coloca en el extremo 3' de $\downarrow z$, añadiendo los nucleótidos complementarios a los de $\uparrow xyz$, para acabar formando la hebra doble $\updownarrow xyz$.

Amplificación mediante PCR Obtención de múltiples copias de una hebra de ADN aplicando la *reacción en cadena de la polimerasa* (del inglés, *Polymerase Chain Reaction - PCR*). El proceso requiere la presencia de cebadores complementarios a los extremos 3' y 5' de todas las hebras presentes en el tubo de ensayo. Para cada hebra doble $\updownarrow xyz$, si los cebadores $\uparrow x$ y $\downarrow z$ están presentes, se obtienen dos

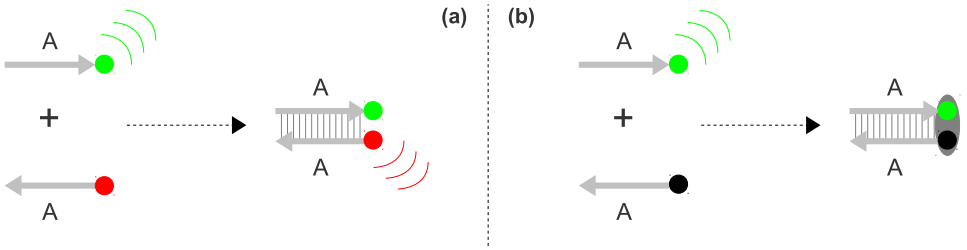

Figura 1.5: Aplicación de técnicas FRET para el marcado de hebras de ácidos nucleicos. (a) Donador y aceptador fluorescentes. (b) Donador fluorescente y aceptador inhibidor.

copias por ciclo. Por tanto, es posible incrementar el número de copias de cada hebra a ritmo exponencial: $2^{número\,de\,ciclos\,de\,PCR}$ copias por hebra.

Transferencia de energía de resonancia de Förster (Del inglés, *Förster resonance energy transfer*) Es un mecanismo transferencia de energía entre cromóforos[1]. Se basa en que la excitación de un cromóforo puede transmitirse a otro más cercano, de manera que pueden tener dos funciones: donador y aceptador. Cuando los cromóforos son fluorescentes de denominan fluorocromos, y pueden utilizarse para etiquetar moléculas de ácidos nucleicos añadiéndolos en sus extremos, con las siguientes aplicaciones:

- *Donador y aceptador con distintos colores de fluorescencia.* Cuando ambos tipos de fluoróforos están separados en el espacio, el donador emite su color de fluorescencia mientras que el aceptador no emite fluorescencia. Sin embargo, cuando se enfrentan los extremos de dos hebras etiquetados con donador y aceptador, se reduce el nivel de fluorescencia del donador mientras que aumenta el nivel de fluorescencia del receptor (ver figura 1.5-a).
- *Donador emite fluorescencia y aceptador no emite fluorescencia.* Cuando el donador está separado del aceptador, emite fluorescencia. Sin embargo, cuando se enfrentan los extremos de dos hebras etiquetados con donador y aceptador, este último absorbe toda la energía del primero, prácticamente eliminándose por completo el nivel de fluorescencia (ver figura 1.5-b). En estos casos, el aceptador se suele denominar con su terminología inglesa: *quencher*.

[1]Sustancia que contiene muchos electrones capaces de absorber energía o luz visible, y excitarse para así emitir diversos colores.

1.3. Organización de la memoria

La computación con ADN es una disciplina científica de carácter multidisciplinar, en la que confluyen campos tan diversos como la algoritmia, la complejidad computacional, la lógica, la biología molecular o la genética. Es por esto que requiere de preparación específica en diversos campos científicos y del establecimiento de relaciones entre expertos en todas estas disciplinas.

Con el propósito de hacer de esta tesis un documento autocontenido, que facilite al lector la comprensión de los distintos modelos de computación biomolecular analizados y presentados, se han incluido en la introducción algunas nociones sobre la estructura básica y propiedades de los ácidos nuleicos (ADN y ARN), así como sus operaciones básicas de manipulación. Si bien para un lector con formación en biología, todos estos conceptos le resultarán muy básicos, resultarán de gran ayuda para introducir la tecnología del ADN recombinante a los lectores sin esta base de conocimientos. De igual manera, en los capítulos 2 y 4 se incluyen revisiones de conceptos sobre lógica proposicional para el lector que no esté familiarizado con la computación lógica.

La memoria se organiza como sigue:

Capítulo 1 Introducción de la memoria. Conceptos básicos sobre el ADN, sus operaciones fundamentales y la tecnología del ADN recombinante.

Capítulo 2 Estudio de los diferentes modelos y técnicas utilizados en computación biomolecular. Desde los trabajos pioneros de Adleman hasta los modelos más recientes de hibridación competitiva, pasando por el modelo de computación autónoma *Whiplash PCR*, autómatas basados en enzimas de restricción y autómatas basados en deoxiribozimas.

Capítulo 3 Contextualización y planteamiento del problema que se pretende resolver: diseño de dispositivos de ADN con aplicación biomédica que implementan paradigmas de computación lógica.

Capítulo 4 Presentación de resultados: Dispositivos para la realización de computación lógica con ADN.

Capítulo 5 Análisis de las aplicaciones, ventajas y desventajas de las soluciones propuestas.

Capítulo 6 Resumen de las soluciones aportadas y conclusiones obtenidas en la Tesis. Concreción de problemas abiertos y todavía no resueltos y especificación de las líneas futuras de investigación.

Bibliografía Material bibliográfico empleado.

Capítulo 2

Computación biomolecular

Este capítulo pretende ser un estudio sobre el estado de la cuestión en computación biomolecular, mostrando la evolución del campo desde los trabajos iniciales de Adleman hasta los recientes modelos de computación con ARN. Una buena parte de esta recopilación ha sido presentada en un artículo para la revista *Current Bioinformatics* (32), publicado en junio de 2010.

2.1. El algoritmo de Adleman

2.1.1. Definición del problema

El algoritmo desarrollado por Adleman (6) resolvía el problema del camino hamiltoniano: dado un grafo dirigido $G = \{V, E\}$ con un conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de aristas $E \subseteq V \times V$ (ver figura 2.1-a), un camino hamiltoniano pasa por cada vértice del grafo dirigido exactamente una vez, comienza en un vértice designado como inicial, $v_{in} \in V$, y finaliza en $v_{out} \in V$. Encontrar un camino de este tipo es un problema NP-completo.

2.1.2. Codificación y funcionamiento

Adleman ideó un esquema de codificación de grafos dirigidos en ADN, de manera que se podía explotar la capacidad de autoensamblado del ADN a partir de la hibridación espontánea de hebras complementarias. Cada vértice v_i (2, 3 y 4 en la figura 2.1-b) se asignaba a una secuencia de ADN, y cada arista e_{ij} (2→3 y 3→4 en la figura 2.1-b) se

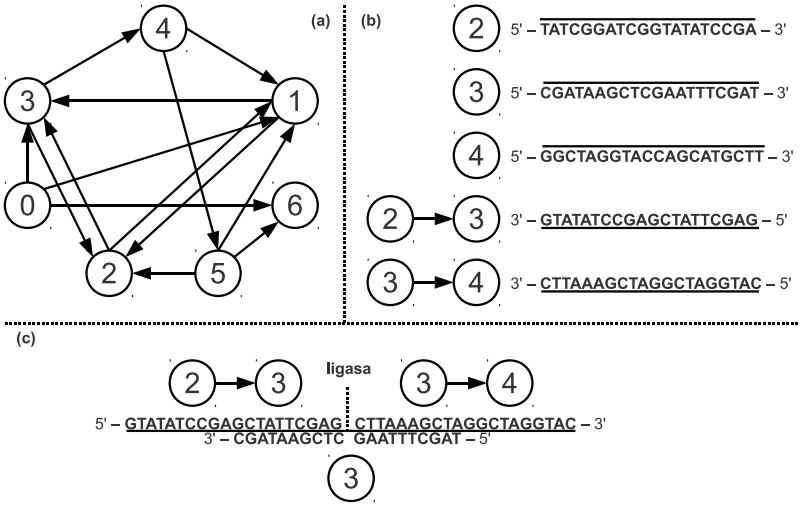

Figura 2.1: El algoritmo de Adleman. (a) Grafo dirigido $G = \{V, E\}$ con un conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de aristas $E \subseteq V \times V$. (b) Codificación de vértices y aristas en ADN. (c) Formación de caminos por concatenación de aristas mediante la enzima ligasa. Adaptada de Sainz de Murieta et al. (32), con permiso.

asignaba a la concatenación de las secuencias complementarias a la mitad 5' de v_i y a la mitad 3' de v_j. Posteriormente se mezclaba un alto número de copias de las secuencias de ADN codificadas (hasta 10^{20}hebras por tubo de ensayo).Las hebras vértice hibridan con las hebras arista gracias a la complementariedad de Watson-Crick y a la función de la enzima ligasa (ver figura 2.1-c), generando un conjunto de caminos potenciales, que con una probabilidad muy alta incluía todas las soluciones posibles. Los siguientes pasos del algoritmo se centran en el filtrado de las hebras de ADN, mediante técnicas de PCR o electroforesis en gel, para obtener el conjunto correcto de soluciones.

2.1.3. Descripción del algoritmo

El algoritmo comprende los siguientes pasos:

1. Generación aleatoria de caminos sobre el grafo, mezclando en un tubo de ensayo un gran número de hebras de ADN que codifican vértices y aristas, como se ha descrito en el párrafo anterior. La presencia de enzimas ligasa permite el ligado de las dobles hebras resultantes formando los caminos candidatos.

2. Eliminación de todas aquellas hebras que no tengan las secuencias v_{in} y v_{out} como

inicio y final, respectivamente. Para ello se realiza una amplificación con PCR, utilizando v_{in} y v_{out} como cebadores.

3. Eliminación de aquellos caminos que no tengan una longitud específica, proporcional al número de vértices: si el grafo tiene 7 vértices, codificados con secuencias de 20 nucleótidos por vértice, mantener sólo las secuencias de 140 nucleótidos de longitud. Se realiza mediante electroforesis en gel.

4. Mantener en el tubo de ensayo sólo aquellos caminos que recorran cada vértice al menos una vez. Para ello se realizan operaciones sucesivas de desnaturalización y separación por afinidad (una por cada secuencia de vértice).

5. Amplificación mediante PCR de las hebras resultantes del paso 4. Si hay hebras que amplificar, entonces existe un camino hamiltoniano.

Todos los pasos tienen complejidad lineal respecto al número de vértices, excepto el número 4 (desnaturalización + separación por afinidad), cuya complejidad es cuadrática respecto al número de vértices. Por tanto, la complejidad general del algoritmo es $O(n^2)$. Sin embargo, el número de hebras necesarias para codificar todos los caminos posibles crece exponencialmente con el tamaño del problema.

2.2. Computación autónoma: Whiplash PCR

Todos los modelos y bioalgoritmos surgidos en la primera etapa de la computación biomolecular, en su mayoría basados en los trabajos de Adelman, adolecían del mismo problema: dado que el programa no estaba almacenado ni codificado dentro de los tubos de ensayo en un sustrato molecular, necesitaban la presencia externa de una persona ejerciendo de "bioprocesador", es decir, ejecutando todas las instrucciones del programa en forma de operaciones de laboratorio.

Con el objetivo de solventar este problema, un grupo de investigación japonés, liderado por Masami Hagiya, presentó el primer modelo de computación biomolecular autónomo con ADN, comúnmente denominado *Whiplash PCR* (33). Con la ayuda de la enzima ADN polimerasa, su modelo conseguía por fin almacenar el programa en un sustrato biomolecular, permitiendo simular autómatas finitos empleando hebras simples

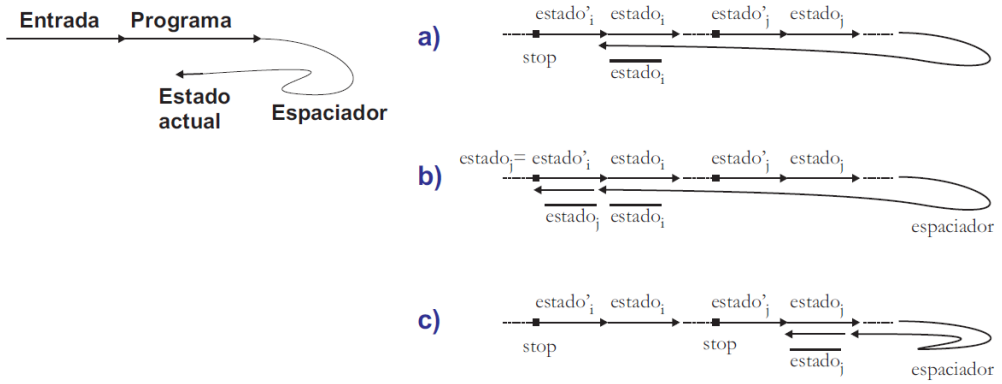

Figura 2.2: Computación autónoma con el método *Whiplash PCR*. Adaptada de Rodríquez-Patón (34), con permiso.

de forma autónoma y aprovechando el paralelismo implícito que ofrece el ADN.

El modelo utiliza una hebra simple de ADN para codificar los posibles movimientos del autómata, de la siguiente manera (ver figura 2.2):

- El extremo 3' de la hebra codifica el estado actual del autómata, como si fuera un cabezal de lectura.
- Las transiciones se codifican a partir del extremo 5' de la hebra, utilizando secuencias de ADN que siguen un patrón $stop - estado'_i - estado_i$, representando la transición del $estado_i$ al $estado'_i$.
- Entre el estado actual (en el extremo 3') y el programa (en el extremo 5') se incluye una secuencia "espaciadora" que no codifica ningún estado. Su objetivo no es otro que facilitar el plegamiento de la hebra y el acercamiento e hibridación del estado actual con los estados del programa.

Al inicio de la ejecución del autómata, el estado actual está compuesto por una secuencia de nucleótidos complementaria a alguno de los estados en el extremo 5'. Bajo las condiciones adecuadas de presión y temperatura, ocurrirá lo siguiente (ver figura 2.2):

- Asumiendo que el estado actual está formado por la secuencia de nucleótidos $\overline{estado_i}$, hibridará con el $estado_i$ de la zona de transiciones, formando una estructura de horquilla (ver figura 2.2-a).

- Posteriormente la enzima ADN polimerasa se fija en el extremo 3' de la hebra, extendiéndola con la secuencia complementaria del $estado'_i$, que pasa a ser el actual (ver figura 2.2-b).

- Cuando el ADN polimerasa llega a la secuencia *stop* de la transición, finaliza su actividad de polimerización.

- Aplicando calor en el tubo se consigue desnaturalizar el estado actual, deshaciendo temporalmente la horquilla.

- Suponiendo que existe otro $estado_j$ en el autómata tal que $estado'_i = estado_j$, tras aplicar un ciclo de enfriamiento en el tubo, el estado actual hibridará con el $estado_j$‘ con una probabilidad muy alta, el ADN polimerasa volverá a extender el extremo 3' y el estado $estado'_j$ pasará a ser el nuevo estado actual (ver figura 2.2-c).

El principal problema que plantea el método es el paso de rehibridación. El estado actual no es el único que puede hibridar con estados de la zona de transiciones. También es posible que alguno de los antiguos estados actuales (originalmente en el extremo 3' pero desplazados por los nuevos estados actuales durante la ejecución del programa) hibriden con alguno de los estados de la zona de transiciones. Favorecer la hibridación del estado actual frente al resto de posibles hibridaciones no es fácil de lograr experimentalmente. Pese a que se propuso una extensión a este método buscando solucionar este problema (35),

2.3. Autoensamblado del ADN

Las técnicas de autoensamblado de ADN pueden verse como una combinación de los conceptos de computación con ADN surgidos del trabajo de Adleman (6) y un modelo formal de embaldosado en 2D desarrollado por Hao Wang (36, 37).

2.3.1. Baldosas de Wang

Un embaldosado es una colección de formas básicas (llamadas *baldosas*) que encajan entre sí en un plano. Las baldosas de Wang son cuadrados de igual tamaño, con una etiqueta o color específicos asociados a cada uno de los lados. Los cuadrados pueden ensamblarse lado con lado, siempre que los lados colindantes de ambos cuadrados tengan la misma etiqueta o color. Wang presentó su algoritmo de embaldosado en 1961, pero no fue hasta 1965 cuando Robert Berger (36) demostró que dicho modelo era equivalente a una máquina de Turing, o lo que es lo mismo, un modelo de computación universal.

2.3.2. Bloques de ADN

Erik Winfree fue pionero en aplicar los modelos computacionales de embaldosado sobre nanoestructuras de ADN, construyendo bloques de ADN a modo de "ladrillos computacionales" capaces de procesar de manera autónoma a medida que se ensamblan con otros ladrillos. La primera demostración experimental de este tipo de computación se logró en 2000 (38), con la implementación de un conjunto de bloques de ADN que al ensamblarse realizaban la operación lógica XOR. Tras este hito se sucedieron diversos trabajos similares para otras primitivas lógicas de computación (39, 40), algunos de ellos enfocándose incluso en la reducción de errores e implementando mecanismos de corrección y autorreparación de los bloques (41, 42).

2.3.3. ADN como elemento de andamiaje: Origami

Dentro de la computación mediante autoensamblado de ADN, hay otra línea de investigación que utiliza hebras de ADN como andamiaje en la construcción de nanoestructuras bidimensionales. El método requiere la combinación de una hebra larga a modo de estructura, de cadena simple, y un conjunto de hebras de corta longitud que definen el patrón de la malla. El primer artículo utilizando hebras de andamiaje se publicó en 1999 (43), al que le sucedió otro interesante trabajo donde se lograba ensamblar un código de barras a partir de una malla de ADN (44). Pero fue en 2006 cuando Paul Rothemund (45) generalizó el modelo de andamiaje denominado *origami*: una hebra principal de ADN, larga, y una gran variedad de hebras auxiliares cortas que funcionan como grapas, cada una de ellas uniendo partes remotas de la hebra principal. Modificando la composición de las hebras auxiliares (las grapas), Rothemund es capaz

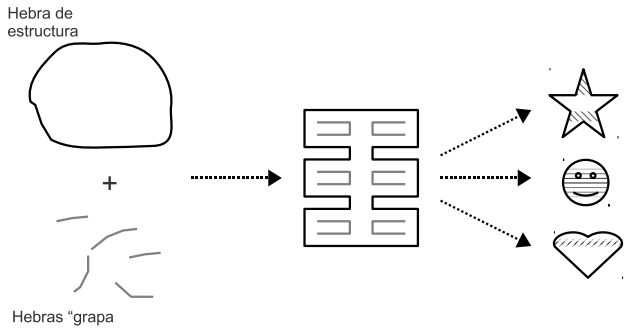

Figura 2.3: Diferentes patrones obtenidos en mallas de ADN utilizando el modelo de andamiaje origami. Adaptada de Sainz de Murieta et al. (32), con permiso.

de producir distintos patrones en la malla resultante, perfectamente visibles mediante AFM (microscopio de fuerza atómica) (ver figura 2.3).

2.4. Autómatas de ADN basados en enzimas de restricción: el autómata de Benenson

Uno de los logros más significativos y de mayor difusión de la computación biomolecular (tras los primeros años enfocados a la resolución de problemas NP-completos) fue el conseguido en 2001 por el equipo de Ehud Shapiro: desarrollaron el primer autómata molecular in vitro (10). Este autómata estaba compuesto por un conjunto de biomoléculas que codificaban las transiciones de estado; con la ayuda de enzimas de restricción y ligasa, el sistema era capaz de procesar un conjunto de entradas biomoleculares y producir una salida, también en formato biomolecular. Posteriormente se presentó una versión estocástica del anterior autómata (12), pero su aplicación más significativa se enfocaba directamente en el ámbito de la biomedicina (11, 13). Las biomoléculas del autómata codificaban los marcadores de diagnóstico de una enfermedad concreta (por ejemplo, cáncer), y su tratamiento. En otras palabras:

1. Un sistema experto de diagnóstico con un conjunto de reglas aplicadas en cascada, como por ejemplo "SI la concentración de ARNm es alta, ENTONCES diagnóstico positivo".

2. El tratamiento correspondiente al diagnóstico positivo, que sólo se liberaba si todos los pasos del proceso de diagnóstico se completaban.

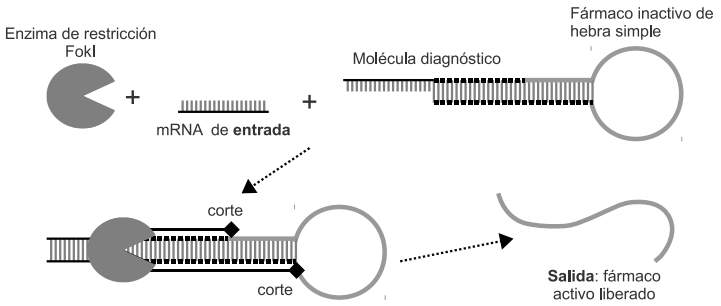

Figura 2.4: Autómata biomolecular a partir de moléculas de ADN y de la enzima de restricción FokI. Adaptada de Sainz de Murieta et al. (32), con permiso.

La figura 2.4 muestra el funcionamiento básico de este autómata: su programa codifica un proceso de diagnóstico en una horquilla de ADN, de forma que la salida (el fármaco inactivado en forma de hebra simple de ADN) es la zona del bucle y las diferentes señales moleculares a comprobar están codificadas en el tronco de la horquilla. La señal activa está codificada en en el extremo 5' protuberante de la horquilla. Cuando una entrada en forma de molécula de ARNm es complementaria al extremo 5' protuberante de la horquilla, ambas moléculas quedan hibridadas. Esto permite que la enzima FokI se posicione en su zona de reconocimiento y corte la molécula asimétricamente. Por consiguiente, el autómata sólo produce una señal de salida en forma de hebra simple de ADN en presencia de una señal (o secuencia de señales), en forma de ARNm de entrada, como consecuencia del diagnóstico positivo.

El modelo computacional que describe este proceso de diagnóstico es una *máquina de estados finitos*, que alterna dos estados: "diagnóstico positivo" y "diagnóstico negativo". El tratamiento sólo se libera cuando el estado del autómata al final del proceso es "diagnóstico positivo".

2.5. Autómatas basados en deoxiribozimas

Al contrario que las enzimas de restricción, que están formadas por proteínas, las deoxiribozimas (también llamadas *DNAzimas*) se componen de moléculas de ADN. Las deoxiribozimas también ejercen una acción catalítica que permite cortar ADN de hebra simple, mientras que las enzimas de restricción cortan ADN en hebra doble.

Una deoxiribozima está formada por una zona catalítica central flanqueada por dos dominios de reconocimiento para un sustrato específico de ADN. El sustrato en cuestión debe incluir un ribonucleótido flanqueado por dos zonas que son complementarias al los dominios de reconocimiento de la DNAzima. Cuando el substrato hibrida con los dominios de reconocimiento, su acción catalítica corta el substrato en la posición donde se encuentra el ribonucleótido (ver figura 2.5-a).

2.5.1. Puertas lógicas y autómatas

En 2002, Stojanovic et al. (22) diseñaron un conjunto de dispositivos capaces de activar y desactivar el funcionamiento de las DNAzimas en función de unas determinadas hebras de entrada, siendo así capaces de implementar puertas lógicas, cuyo valor de verdad en la salida es "cierto" cuando la acción catalítica consigue cortar el sustrato, y "falso" en caso contrario. Dichos dispositivos presentaban distintas configuraciones:

- *Puerta NOT.* Esta variante incorpora una horquilla en la zona catalítica central. En presencia de una hebra de ADN de entrada, cuya secuencia de nucleótidos es complementaria al bucle de la horquilla, se producirá una hibridación entre ambas que hará perder a la DNAzima su conformación estructural, quedando desactivada su acción catalítica. De esta manera, se consigue inhibir el corte del sustrato. Al dar una respuesta positiva (corte del sustrato) en ausencia de una señal de entrada, y una respuesta negativa (inhibición corte del sustrato) en presencia de una señal de entrada, este modelo implementa una puerta lógica NOT (ver figura 2.5-b).

- *Sensor simple.* Esta variante incorpora una horquilla haciendo coincidir su tronco con uno de los dominios de reconocimiento de la DNAzima, inhibiendo la hibridación del sustrato catalítico y por consiguiente su corte. En presencia de una determinada señal de entrada, complementaria al bucle de la horquilla, se abren los brazos de su tronco permitiendo la hibridación y el corte del sustrato. Al ofrecer una respuesta positiva (corte del sustrato) en presencia de una señal de entrada concreta, y una respuesta negativa (inhibición del corte del sustrato) en caso contrario, este modelo implementa un sensor simple (ver figura 2.5-c).

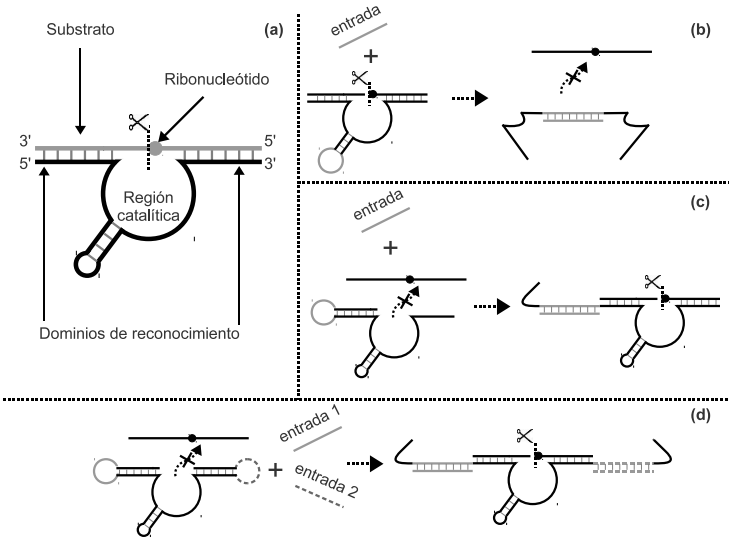

Figura 2.5: Puertas lógicas a partir de deoxiribozimas. (a) Estructura de una deoxiribozima. (b) Puerta lógica NOT. (c) Sensor simple. (d) Puerta lógica AND. Adaptada de Sainz de Murieta et al. (32), con permiso.

- *Puerta AND.* Esta variante añade dos horquillas, una sobre cada dominio de reconocimiento de la DNAzima. Dichos dominios sólo quedarán hibridados en presencia de dos señales de entrada distintas, complementarias a sendos bucles de horquilla. Al ofrecer una salida positiva (corte del sustrato) únicamente en presencia de las dos señales de entrada esperadas, y una salida negativa (inhibición del corte del sustrato) en cualquier otro caso, este modelo implementa el funcionamiento de una puerta lógica AND (ver figura 2.5-d).

En un trabajo posterior (23), estas puertas lógicas se utilizan para implementar un autómata biomolecular que codifica una versión 3 x 3 del juego tic-tac-toe (tres en raya). Dicho autómata es capaz de competir de manera interactiva con un oponente humano, marcando sus movimientos mediante la emisión de fluorescencia. La estrategia del autómata es perfecta: siempre logra ganar o empatar. En un trabajo mucho más reciente (24), el mismo equipo ha implementado un juego similar llamado *tit-for-tat*, con una interesante innovación: la estrategia de respuesta del autómata puede ser configurada antes del juego, mediante la utilización de un conjunto de moléculas de entrenamiento.

Uno de los principales inconvenientes de este tipo de puertas lógicas radica en la diferencia de formatos entre las señales de entrada y las de salida. Esto dificulta su utilización en cascada, donde la salida de una puerta se realimenta como entrada en otra puerta diferente. Con la intención de solventar esta limitación, Stojanovic et al. (25) presentaron otro trabajo expandiendo el anterior conjunto de puertas lógicas, añadiendo ligasa y estructuras de ADN que facilitan la operación de ligado. De esta forma, distintas hebras de salida pueden combinarse para formar hebras de ADN más largas, adquiriendo el formato adecuado para funcionar como hebra de entrada de otra puerta diferente. Estos nuevos componentes también pueden utilizarse para regenerar los substratos ya cortados por una DNAzima, retornando así el circuito en su estado inicial. En otro reciente trabajo presentado por Elbaz et al. (27), se presenta una plataforma computacional construida a partir de un conjunto de módulos (fragmentos de DNAzimas y substratos). La plataforma logra implementar un conjunto universal de puertas lógicas que pueden ser aplicadas en cascadas de varios niveles.

2.5.2. Motores moleculares

Las deoxiribozimas también se han utilizado en un contexto diferente al de la computación con ADN, como son los *motores moleculares*. En Stojanovic et al. (26), se acoplan cuatro DNAzimas mediante una molécula de streptavidina, formando un dispositivo de apariencia similar a una araña con cuatro patas. Dicho dispositivo se coloca en una matriz cubierta de substratos que permiten la hibridación de las "patas" de los dispositivos. Una vez que la DNAzima corta el substrato, la pata se libera y queda lista para hibridar con otro substrato diferente. De este manera, la araña implementa un dispositivo de movimiento aleatorio "hacia adelante" que nunca puede volver atrás en sus pasos, ya que todos los puntos de anclaje anteriores han sido deshabilitados. En un trabajo más reciente (46), estas arañas moleculares logran elegir direcciones en sus movimientos, detectando y modificando distintos tipos de substratos integrados en una malla bidimensional de ADN Origami.

2.6. Hibridación competitiva

La hibridación competitiva del ADN es otra línea de investigación de gran importancia en computación biomolecular, denominada en inglés *DNA strand displacement*.

La técnica se introdujo por primera vez en Yurke et al. (14), con el desarrollo de unas "pinzas moleculares". Pero fue seis años después, tras el trabajo de Seelig et al. (15), cuando la técnica comenzó a utilizarse ampliamente para el diseño de dispositivos de detección y procesado de diferentes tipos de moléculas de ácidos nucleicos (ADN o ARN). El mecanismo básico de estos sensores se denomina *hibridación competitiva mediante extremos cohesivos* (en inglés, *toehold-mediated strand displacement*) y está descrito en la figura 2.6-a. El dispositivo de dicho ejemplo se compone de dos hebras de ADN parcialmente hibridadas A y B, que detectan la presencia de una hebra de entrada A. A y A' son totalmente complementarias. Si la entrada A está presente, desplaza B del complejo BA debido a la mayor atracción de A hacia A' y la mayor estabilidad de AA' en comparación con BA. La región de A que queda sin hibridar en el complejo BA y su región complementaria en la hebra A' se denominan comúnmente con su término en inglés: *toeholds*. Los toeholds juegan un papel fundamental en el proceso de hibridación competitiva, ya que funcionan tanto como zona de reconocimiento como de punto de anclaje.

La principal limitación de la hibridación competitiva mediante extremos cohesivos es su irreversibilidad: una vez que se libera la salida B y se forma el complejo AA', la tasa de la reacción inversa (disociación de AA' e hibridación de A y B) es prácticamente nula, ya que . En trabajos posteriores (16, 17), se ha evolucionado el modelo para permitir reversibilidad en este tipo de reacciones, dando lugar al modelo denominado *intercambio de extremos cohesivos* (en inglés, *toehold exchange*). Si observamos el ejemplo de la figura 2.6-b, veremos que este método es similar al del ejemplo anterior en cuanto a que una hebra simple se une al extremo cohesivo de una molécula de doble hebra, iniciando un proceso de hibridación competitiva. La diferencia fundamental entre ambos procesos radica en que la hebra simple C es más corta que su equivalente A en el ejemplo anterior. Una vez que C se ha unido al extremo cohesivo libre en A y cada nucleótido de C está apareado con A, la hebra B va a seguir unida a A, ya que C no es suficientemente larga para desplazarla totalmente. El desplazamiento de B ocurrirá como consecuencia de una reacción espontánea de disociación, siempre y cuando la longitud de la región que une A y B en este punto sea suficientemente corta.

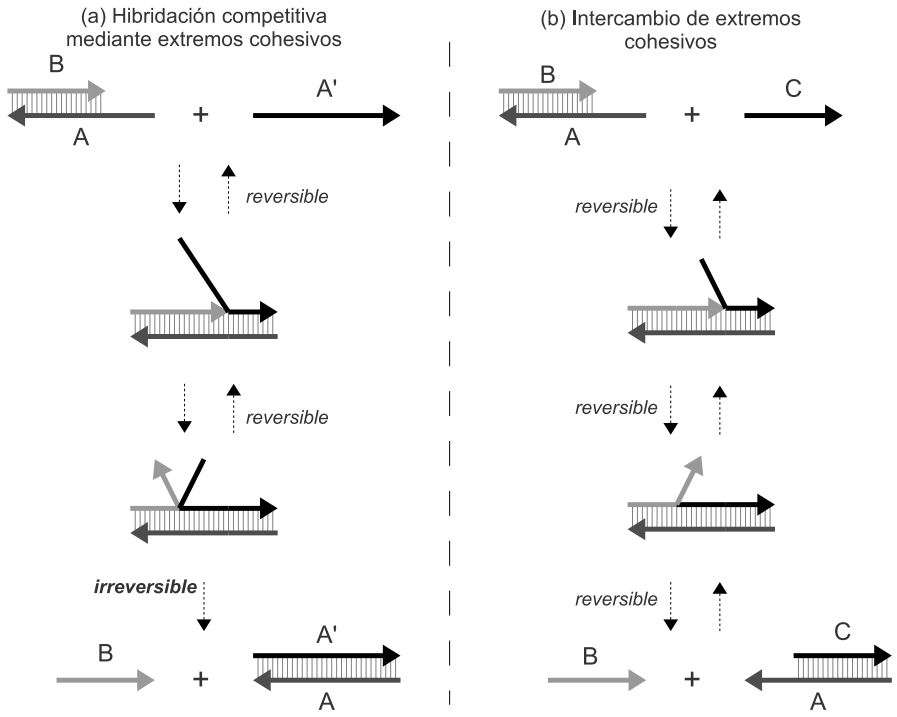

Figura 2.6: (a) Hibridación competitiva mediante extremos cohesivos. (b) Intercambio de extremos cohesivos. Adaptada de Sainz de Murieta et al. (32), con permiso.

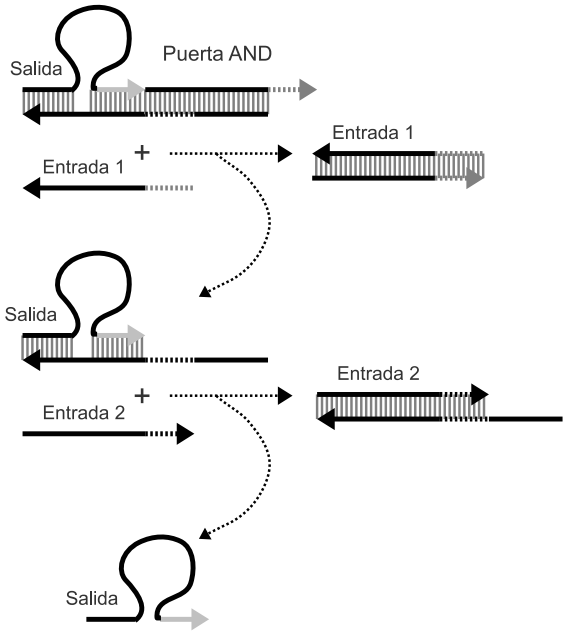

Figura 2.7: Implementación de una puerta AND mediante hibridación competitiva. Adaptada de Sainz de Murieta et al. (32), con permiso.

La técnica de la hibridación competitiva del ADN ha dado lugar a la implementación de diversos modelos de puertas lógicas (15, 47, 48). La figura 2.7 describe un modelo de puerta AND incluida en Seelig et al. (15), trabajo pionero en la implementación de puertas lógicas mediante hibridación competitiva. Está formada por tres hebras de ADN: la hebra de salida (con un bucle) y dos hebras cuyas secuencias de nucleótidos son complementarias a las hebras de entrada. Cuando la entrada 1 se aproxima a la puerta, ambas se hibridan sus extremos cohesivos o toeholds (representados en la figura 2.7 con línea discontinua gris), iniciando un proceso de hibridación competitiva que finaliza con la liberación de una de las hebras de la puerta, destapándose un nuevo toehold (representado en la figura 2.7 con línea discontinua negra). Llegados a este punto, si la entrada 2 está presente, hibrida con el toehold recién descubierto en la puerta, iniciándose un segundo proceso de hibridación competitiva que termina liberando la salida de la puerta.

La hibridación competitiva se ha convertido en una técnica ampliamente extendida en el diseño de cualquier tipo de circuito lógico que pretenda aprovechar la dinámica de las reacciones entre ácidos nucleicos:

- Soloveichik et al. (18) han desarrollado un modelo capaz de aproximar el comportamiento de sistema dinámicos de reacciones químicas mediante un conjunto de reacciones sencillas de hebras de ADN.

- Cardelli (19) ha modelado un álgebra que describe formalmente las hebras de ADN y la operación de hibridación competitiva. Este modelo proporciona a los investigadores en computación con ADN una herramienta para realizar operaciones con ADN fuera del laboratorio. El modelo ha evolucionado hacia un lenguaje de programación que implementa dicha álgebra *in silico*, usando un simulador de hibridación competitiva (20).

- Después de las "pinzas" implementadas por Yurke et al. (14), se han presentado un buen número de robots moleculares (49, 50, 51). Más recientemente se ha presentado una evolución muy interesante de todos esos modelos (52), donde el robot no sólo es capaz de moverse sobre una malla de ADN, sino que también es capaz de capturar, transportar y soltar nanopartículas de oro entre distintos contenedores.

2.7. Computación lógica con ADN

2.7.1. Introducción a la lógica proposicional

Un programa lógico (53, 54) es un conjunto finito de hechos y reglas que, a partir de un conjunto de premisas (axiomas o condiciones entrada) y aplicando un conjunto de inferencia, es capaz de deducir una o más conclusiones (consecuencias lógicas o condiciones de salida). Este proceso de razonamiento, denominado *inferencia*, es la piedra angular de la programación lógica.

Al referirnos a hechos utilizaremos el término *proposición*, que determina la mínima entidad sintáctica en lógica proposicional. Las proposiciones representan enunciados afirmativos simples, que en adelante representaremos con una letra. Por ejemplo, $P =$ "Sócrates está presente" o $Q =$ "hebra de ADN presente". Cada proposición puede tomar los valores "cierto" o "falso". Representaremos $P =$ "cierto" como P y $P =$ "falso" como $\neg P$.

Las reglas de implicación establecen relaciones entre proposiciones de entrada (premisas) y las proposiciones de salida (conclusiones). Se pueden representar como reglas de tipo "SI-ENTONCES". Por ejemplo, el enunciado "SI P ENTONCES Q" es una implicación, lo que significa que si P es cierto, entonces Q debe ser también cierto. En lógica, las reglas de implicación se representan don una flecha que va desde la premisa a la conclusión, por lo que podemos escribir la anterior regla como $P \rightarrow Q$.

Tanto premisas como conclusiones pueden estar formadas por más de una proposición. Por ejemplo, la premisa de la regla de implicación "SI P Y Q ENTONCES R" está compuesta por dos proposiciones (P y Q). Significa que, si las proposiciones P y Q son ciertas, entonces R también es cierto. En lógica, la conectiva "Y" se representa con el símbolo $\wedge$, así que la anterior implicación puede reescribirse como $P \wedge Q \rightarrow R$. Sin embargo, en la regla de implicación "SI P ENTONCES Q O R", es la conclusión la que está formada por varias proposiciones (Q y R). Significa que, si la proposición P es cierta, entonces, o bien Q o R es también cierta (o ambas). En lógica, la conectiva "O" se representa con el símbolo $\vee$, así que la anterior implicación puede reescribirse como $P \rightarrow Q \vee R$.

Las dos reglas de inferencia más ampliamente utilizadas se denominan *modus ponens* y *modus tollens*. La regla modus ponens dice que, a partir de un hecho P y una implicación $P \rightarrow Q$, se deduce Q. Formalmente se puede expresar así: $(P \rightarrow Q) \wedge P \rightarrow Q$. La regla modus tollens dice que, a partir de un hecho $\neg Q$ y una implicación $P \rightarrow Q$, se deduce $\neg P$. Se puede expresar formalmente así: $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$. Como ejemplo, tomemos la regla de implicación $P \rightarrow Q$ como nuestro programa lógico. $P \rightarrow Q$ representa el enunciado "SI la hebra P de ADN está presente ENTONCES enfermedad Q presente" (o hebra Q de ADN $\rightarrow$ enfermedad Q). Si añadimos la proposición P, "hebra P de ADN presente" como entrada del programa, se dispara la regla de inferencia modus ponens, infiriendo la proposición Q ="enfermedad Q presente" como salida. Por el contrario, si añadimos la proposición $\neg Q$ ="enfermedad Q no presente" como entrada del programa, se dispara la regla de inferencia modus tollens, infiriendo la proposición $\neg P$ ="hebra de ADN P no presente" como salida.

2.7.2. Inferencia con moléculas de ADN

Existen algunos precedentes en el uso de moléculas de ADN en la implementación de computación lógica, bien utilizando el autoensamblado de ADN y la reacción en cadena de la polimerasa (PCR) para implementar funciones booleanas (55) e inferencia lógica (56), bien utilizando horquillas de ADN para representar μ-fórmulas (33), interrogaciones lógicas (57), o inferencia entre cláusulas de Horn (58). Pero fue el trabajo presentado en Ran et al. (59) el que reavivó el interés y redescubrió el potencial de aplicar el paradigma de computación lógica en computación biomolecular. Basándose en trabajos previos sobre el autómata de Benenson (10, 11, 12, 13), desarrollaron un sistema capaz de realizar deducciones lógicas simples con moléculas de ADN. En dicho trabajo, las proposiciones y las reglas de implicación se codifican utilizando moléculas de ADN de doble hebra con un extremo cohesivo libre (ver figura 2.8). La secuencia de nucleótidos de los extremos cohesivos codifica de manera única una proposición de entrada (en la figura 2.8, $P = síntoma_1$"). Cuando una proposición y una implicación tienen extremos libres complementarios, ambas moléculas hibridan en una. Posteriormente la enzima Fok I corta la molécula resultante en dos nuevas partes, diferentes de la proposición e implicación iniciales. Al hibridarse una de estas partes con una hebra de ADN auxiliar, la molécula resultante codifica la conclusión inferida de la proposición e implicación iniciales. Dicha conclusión puede utilizarse como entrada para otro proceso de inferencia aplicado en cascada, o leerse como la salida del sistema utilizando técnicas FRET (ver figura 2.8).

2.8. Computación con ARN

Los últimos 10 años han sido bastante prolíficos en lo que ser refiere a la investigación con ARN. Desde que Fire y Mello descubrieron el ARN de interferencia en 1998 (hecho por el que se les galardonó con el premio Nobel en 2006) (60), se han descubierto nuevos tipos de moléculas de ARN endógeno, como son el ARN interferente pequeño (ARNip), micro-ARN, ARN antisentido, ribozimas, riboswitches o riboreguladores. Estas moléculas presentan nuevas propiedades de expresión genética, que permiten explicar procesos relacionados con el silenciamiento de genes específicos. En la actualidad se están utilizando diversas moléculas relacionadas con el ADN de interferencia como componentes

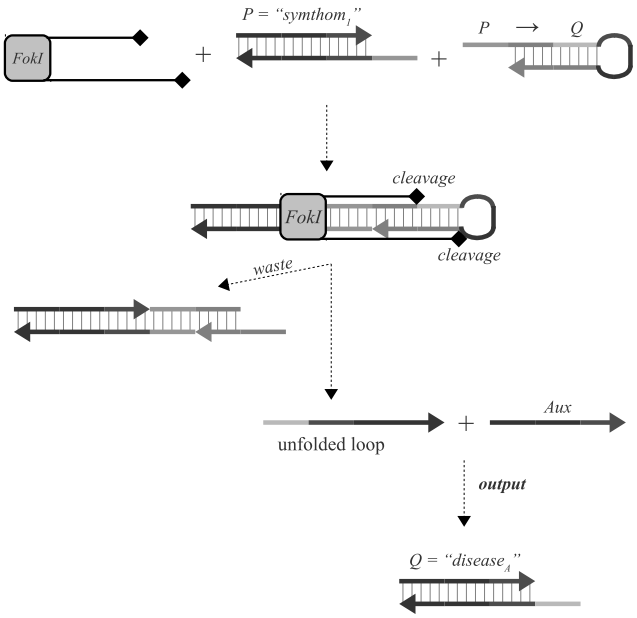

Figura 2.8: Deducciones lógicas simples con moléculas de ADN. La figura describe el modo de funcionamiento básico del sistema de inferencia desarrollado en Ran et al. (59).

de fármacos en ensayos clínicos.

El interés en el ARN de interferencia también ha llegado hasta la computación biomolecular. Existen nuevos diseños de procesadores basados en ARN pueden detectar, procesar y generar como salida nuevas moléculas de ARN.

- Miró-Bueno et al. (31) desarrollaron en 2009 un dispositivo sintético biomolecular que genera señales oscilatorias a partir de dos genes que interactúan entre sí y una ribozima.

- Win et al. (30) desarrollaron en 2008 un sistema ensamblado mediante dispositivos de ARN (riboswitches y ribozimas) que implementa *in vivo* tanto puertas lógicas como filtros de señal que muestran un comportamiento cooperativo. Los riboswitches se unen a pequeños ligandos con el objetivo de regular la expresión genética mediante cambios conformacionales, y las ribozimas son enzimas compuestas por ARN que catalizan cortes en el ARN. Un trabajo del mismo equipo publicado en 2010 (29), ha logrado modificar el proceso de splicing del ADN utilizando dispositivos de ARN. Mediante la detección del exceso de ciertas proteínas,

estos dispositivos logran la traducción de genes que quedarían silenciados sin su operación.

- Rinaudo et al. (28) desarrollaron en 2007 un circuito lógico en ADN capaz de procesar y analizar hasta cinco entradas distintas de ARNip. Dicho ARNip se sintetizó en forma de hebra doble y su objetivo era reprimir la traducción de un gen específico.

- En una reciente colaboración entre Yaakov Benenson (ETH Zürich) y Ron Weiss (MIT), se ha desarrollado un circuito sintético de regulación transcripcional, capaz de clasificar distintos tipos de células en función de los distintos tipos de micro-ARN presentes en cada una de ellas, y desencadenar una respuesta específica si los distintos niveles de micro-ARN corresponden a un patrón objetivo. Como prueba de concepto, su sistema ha sido capaz de clasificar selectivamente células de cáncer cervical y desencadenar en ellas procesos de apoptosis[1], sin afectar a las células sanas (61).

[1] Proceso de muerte celular regulado genéticamente.

Capítulo 3

Planteamiento del problema

La resolución del *Problema del Camino de Hamilton* presentada por Adleman en 1994 (6) despertó enormes expectativas. En un entorno en el que el precio de síntesis del ADN ya estaba descendiendo a un ritmo vertiginoso, ser capaz de utilizar esta molécula para resolver un problema NP-completo en tiempo polinómico supuso un gran hito en la comunidad científica. Tras la posterior resolución de otros problemas NP-completos (7, 8, 9) aprovechando el paralelismo masivo del ADN, pronto se comprendieron las limitaciones de aplicar un modelo de fuerza bruta como el de Adleman en la resolución de problemas NP-completos. El coste de resolver los problemas en tiempo polinómico suponía un consumo exponencial de espacio (o número de moléculas).

A partir de entonces, la computación biomolecular comenzó a enfocarse en la resolución de problemas de interés biomédico o con aplicación directa en la nanotecnología. Las técnicas de autoensamblado, que se habían estado empleando en la resolución de problemas NP-completos, se recondujeron hacia la actualmente denominada *nanotecnología estructural con ADN*, con numerosas aplicaciones en la producción de biomateriales o como elemento de ensamblado para nanomateriales como partículas de oro o nanotubos de carbono (*nanotecnología de interfaz con ADN*).Pero los modelos que más se han relacionado con la computación biomolecular han tratado de aprovechar los cambios conformacionales que experimentan las moléculas de ADN para llegar a un estado de equilibrio. Dichos modelos se engloban en la *nanotecnología dinámica con ADN*, incluyendo a los autómatas basados en enzimas de restricción, en ribozimas o deoxiribozimas,

así como los dispositivos que funcionan mediante la hibridación competitiva del ADN.

La *hibridación competitiva del ADN* es una de las áreas, dentro de la nanotecnología dinámica con ADN, que más se ha desarrollado en los últimos años. En el capítulo 2 se han incluido múltiples ejemplos de dispositivos con aplicaciones biomédicas implementados utilizando esta técnica, como circuitos de ADN, biosensores o motores moleculares (14, 15, 16, 17, 18, 49, 50, 51). A pesar de los numerosos diseños de dispositivos de ADN funcionando mediante hibridación competitiva hasta la fecha, sorprendentemente ninguno de ellos ha estado dedicado explícitamente a la implementación de reglas de inferencia lógica, como lo son *modus ponens*, *modus tollens* o *resolución*. Tras la publicación en 2009 del estupendo trabajo de Shapiro et al. (59), desde nuestro grupo de investigación pensamos que implementación de reglas de inferencia utilizando la hibridación competitiva del ADN podía dar lugar a dispositivos con una gran variedad de aplicaciones, como por ejemplo:

- Sensores de ADN capaces de identificar combinaciones complejas de secuencias de ácidos nucleicos.
- Fármacos inteligentes, que liberan una determinada molécula tras haber identificado una secuencia de ácidos nucleicos.
- Dispositivos modulares de proceso de información que puedan utilizarse como componentes básicos para la construcción de circuitos de ácidos nucleicos.
- Bases de datos *in vitro* que permitan almacenar registros de una base de datos mediante un sustrato biológico, interrogar, procesar y recuperar esta información.
- Reguladores post-transcripcionales capaces de controlar o modular la expresión de determinados genes.

Los dispositivos que se presentan en el capítulo 4 pretenden dar soluciones a las aplicaciones que se acaban de enumerar.

Capítulo 4

Dispositivos de computación lógica con ADN

El objetivo fundamental de este trabajo ha sido doble: por un lado, utilizar la hibridación competitiva del ADN, muy versátil y extensamente estudiada; por otro lado, tratar de implementar reglas de inferencia de la manera más simple posible. Con esta motivación, hemos diseñado tres modelos sencillos con que implementan reglas de inferencia con ácidos nucleicos.

Entre las aplicaciones de estos dispositivos se incluyen el diagnóstico *in vitro*, el diseño de biosensores inteligentes y la construcción de bases de datos con registros en formato biomolecular que faciliten el análisis genómico.

Los resultados expuestos en este capítulo forman parte de una solicitud de patente de la que la UPM es titular. El documento se presentó en la Oficina Española de Patentes y Marcas (OEPM) el 28 de mayo de 2010 (número de solicitud P201000694), y todavía está pendiente de resolución. Posteriormente, algunos de los dispositivos han sido presentados en congresos internacionales (21, 62).

4.1. Conceptos básicos de lógica proposicional

Aunque algunos de estos conceptos ya han sido adelantados en el capítulo 2, conviene aclarar la terminología que se va a utilizar sobre lógica proposicional antes de proseguir con la descripción de los dispositivos:

- *Proposición (o variable proposicional).* Mínima entidad sintáctica representable en lógica proposicional. Una proposición P puede tomar los valores de verdad *cierto* o *falso*.

- *Asignación de verdad.* Asignación de un valor de verdad a una proposición. $P =$ "*cierto*" representa la asignación del valor de verdad *cierto* a la proposición P; puede representarse de manera simplificada como P. $Q =$ "$falso$" representa la asignación del valor de verdad *falso* a la proposición Q; puede representarse de manera simplificada como $\neg Q$.

- *Fórmula.* se define recursivamente de la siguiente manera:

  - Cada proposición es una fórmula en sí misma.
  - Si F es una fórmula, su negación, $\neg F$, también es una fórmula.
  - Si F y F' son fórmulas, y $\bullet$ es un operador lógico binario, entonces $F \bullet F$' también es una fórmula. El operador binario $\bullet$ hace referencia al operador de conjunción ($\wedge$), disyunción ($\vee$), implicación ($\rightarrow$) o doble implicación ($\leftrightarrow$).

- *Cláusula.* Fórmula que relaciona un conjunto de proposiciones mediante el operador de disyunción ($\vee$).

- *Forma normal conjuntiva (FNC).* Una fórmula está definida en forma normal conjuntiva si está expresada como una conjunción de cláusulas.

- *Regla.* Fórmula que relaciona un antecedente y un consecuente mediante el operador binario de implicación ($\rightarrow$). Cuando el antecedente y el consecuente son proposiciones, hablamos de reglas simples. De lo contrario, hablamos de reglas compuestas.

- *Regla de inferencia.* Modelo de razonamiento que nos permite concluir fórmulas a partir de un conjunto inicial de fórmulas.

  - *Modus Ponens.* Si tenemos la regla simple $P \rightarrow Q$ y la proposición P, entonces podemos concluir Q.
  - *Modus Tollens.* Si tenemos la regla simple $P \rightarrow Q$ y la proposición $\neg Q$, entonces podemos concluir $\neg P$.

- *Resolución.* Si C y C' son cláusulas y P es una proposición, entonces cualquier asignación de valores que satisfaga $C \vee P$ y $C' \vee \neg P$, satisface también $C \vee C$'.
- *Silogismo hipotético.* Si tenemos la reglas simples $P \rightarrow Q$ y $Q \rightarrow R$, entonces podemos deducir la regla $P \rightarrow R$.
- *Refutación.* Una refutación de una fórmula en FNC no satisfacible, φ, es una secuencia de cláusulas $C_1 \ldots C_s$, en las que cada C_i es una cláusula de φ o se infiere a partir de cláusulas anteriores aplicando la regla de resolución, y C_s es la cláusula vacía. Si se puede derivar una refutación a partir de un fórmula inicial, entonces dicha fórmula es insatisfacible.

4.2. De la lógica proposicional a dispositivos de ADN

Los dos primeros dispositivos, que se analizarán en detalle en apartados posteriores, permiten la implementación de inferencia lógica entre reglas (simples o compuestas) y proposiciones, empleando las siguientes reglas de inferencia:

- *Modus ponens.* Cuando una hebra que codifica a una proposición P interacciona con un dispositivo de hebras que codifica la regla $P \rightarrow Q$, libera como salida una hebra que codifica la proposición Q.
- Modus tollens: Cuando una hebra que codifica a una proposición $\neg Q$ interacciona con un dispositivo de hebras que codifica la regla $P \rightarrow Q$, libera como salida una hebra que codifica la proposición $\neg P$.

Ambos dispositivos pueden presentar los siguientes modos de funcionamiento:

- *Básico.* Funcionamiento del dispositivo aislado, sin interactuar con otros dispositivos similares.
- *Iterativo.* Funcionamiento en cascada. Necesita al menos dos dispositivos, de manera que la salida del primero se convierte en la entrada del segundo.
- *Recursivo.* Funcionamiento en composición (anidado). También necesita al menos dos dispositivos, de manera que el segundo es la salida del primero.

El tercer dispositivo está orientado a otro tipo de reglas de inferencia. A diferencia de *modus ponens* y *modus tollens*, que están orientados a la interacción de una entrada (P) y un dispositivo ($P \to Q$), las reglas de inferencia que implementa el dispositivo 3 están orientadas a la interacción entre dispositivos:

- *Resolución.* Sean P y $\neg P$ proposiciones, C y C' cláusulas; cuando dos dispositivos que representan las cláusulas $P \vee C$ y $\neg P \vee C$' interaccionan entre sí, se combinan formando un dispositivo equivalente al que codifica la cláusula $C \vee C$'.

- *Silogismo hipotético.* Sean P, Q y R proposiciones; cuando dos dispositivos que representan las cláusulas equivalentes a $P \to Q$ y $Q \to R$ interaccionan entre sí, se combinan formando un dispositivo equivalente al que codifica la cláusula $P \to R$.

El dispositivo 3 también permite la evaluación de fórmulas lógicas expresadas en forma normal conjuntiva, o lo que es lo mismo, como una conjunción de cláusulas. Para ello necesitará utilizar una hebra de ADN adicional, además de la colocación estratégica de un fluoróforo y una partícula atenuadora de luminosidad (en inglés, *quencher*). En su estado inicial, el dispositivo emite luminosidad. Pero tras introducir una asignación de verdad al sistema, si esta satisface la fórmula, se produce una reacción de hibridaciones en cadena que consigue acercar un quencher a cada fluoróforo, logrando así eliminar completamente la luminosidad del sistema. Si la asignación de verdad no satisface la fórmula el sistema todavía muestra luminosidad.

4.3. Codificación de las proposiciones

Los tres modelos de dispositivos que se presentan en este capítulo representan las variables proposicionales y sus posibles valores de verdad codificándolas mediante hebras de ácidos nucleicos. Dada una proposición P, se le asigna una hebra de ácido nucleico que comprende una secuencia única de nucleótidos. Dicha secuencia está dividida en dos segmentos de igual longitud, que denotamos p_1 y p_2, orientados de manera que el segmento con subíndice 1 es el más cercano al extremo 5' de la hebra, y el segmento con subíndice 2 es el más cercano al extremo 3'. Para representar la negación de una proposición dada, denotada como $\neg P$, se le asigna una hebra de ácido nucleico que comprende una secuencia de nucleótidos de la misma longitud que P y totalmente

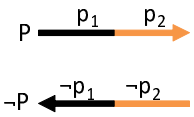

Figura 4.1: Sistema de representación que codifica proposiciones mediante hebras de ácidos nucléicos.

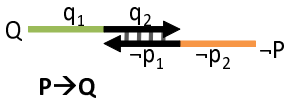

Figura 4.2: Estructura del dispositivo 1 para representación de reglas lógicas.

complementaria[1], que resulta en la formación de hebras dobles de ácidos nucleicos). También la secuencia de nucleótidos que codifica a $\neg P$ está dividida en dos segmentos de igual longitud, denotados por $\neg p_1$ y $\neg p_2$, orientados de manera que el segmento con subíndice 1 es el más cercano al extremo 3', y el segmento con subíndice 2 es el más cercano al extremo 5'. La Figura 4.1 describe el modelo de representación utilizado para codificar las proposiciones.

4.4. Dispositivo 1: inferencia con hibridación competitiva en un paso

El primer dispositivo presentado en este capítulo comprende dos hebras de ácidos nucleicos que codifican dos proposiciones, de acuerdo a la notación de subíndices ya expuesta. Ambas hebras están parcialmente hibridadas entre sí en uno de sus segmentos, de tal manera que los subíndices de los segmentos hibridados no coinciden entre sí; por consiguiente, los subíndices de los dos segmentos libres de hibridación tampoco son coincidentes. El objetivo de este dispositivo es la representación de una regla lógica simple, que genéricamente podemos denotar como $P \rightarrow Q$, y queda relacionada con el dispositivo de la siguiente manera: la hebra cuyo segmento de subíndice 1 está hibridado, codifica a la negación del antecedente ($\neg P$), mientras que la otra hebra codifica al consecuente (Q). La Figura 4.2 describe la estructura del dispositivo 1.

[1] Complementariedad de Watson y Crick

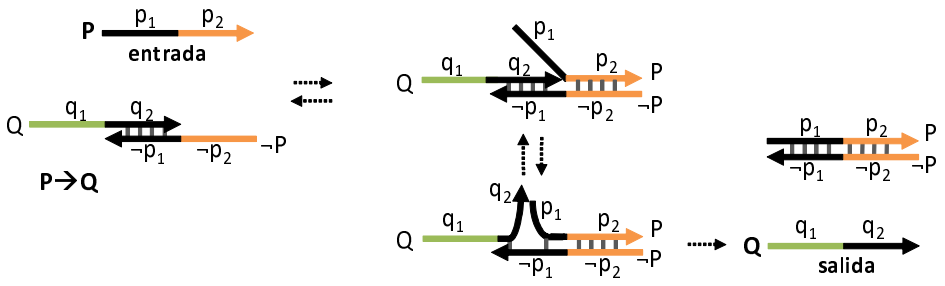

Figura 4.3: Utilización de la regla de inferencia Modus Ponens con el dispositivo 1 en su funcionamiento básico

4.4.1. Funcionamiento básico

Un dispositivo en su estado inicial, que codifique la regla simple $P \rightarrow Q$ según la estructura descrita en la Figura 4.2, variará su configuración en presencia de dos posibles hebras de entrada: ante la aparición del antecedente P, el dispositivo funcionará de acuerdo a la regla de inferencia Modus Ponens, liberando como salida la hebra que codifica a Q. Por el contrario, ante la aparición de la negación del consecuente, $\neg Q$, el dispositivo funcionará de acuerdo a la regla de inferencia Modus Tollens, liberando como salida la hebra que codifica a $\neg P$.

- *Modus Ponens*. El proceso está descrito en la Figura 4.3: ante la entrada de una hebra codificando la proposición P, ésta hibrida su segmento p_2 con el segmentos $\neg p_2$, que está libre en el dispositivo; posteriormente P va a ir ganando posiciones hibridándose con más posiciones de $\neg P$ y desplazando a Q en su segmento q_2, hasta que finalmente P y $\neg P$ formen una hebra doble y liberen la salida del dispositivo: la hebra que codifica a Q.

- *Modus Tollens*. El proceso está descrito en la Figura 4.4: ante la entrada de una hebra codificando la proposición $\neg Q$, ésta hibrida en su segmento $\neg q_1$ con el segmento q_1, que está libre en el dispositivo; posteriormente $\neg Q$ va a ir ganando posiciones hibridándose con más posiciones de Q y desplazando a $\neg P$ en su segmento $\neg p_1$, hasta que finalmente Q y $\neg Q$ formen una hebra doble y liberen la salida del dispositivo: la hebra que codifica a $\neg P$.

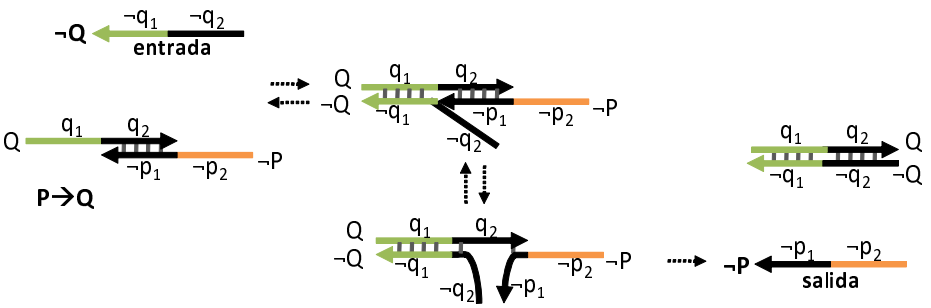

Figura 4.4: Utilización de la regla de inferencia Modus Tollens con el dispositivo 1 en su funcionamiento básico.

4.4.2. Funcionamiento iterativo

Dado que el dispositivo utiliza como entrada y salida hebras simples de la misma longitud, es posible disponer varios de ellos formando cascadas de inferencia. De esta manera, podemos tener dos reglas simples, $P \rightarrow Q$ y $Q \rightarrow R$, de forma que ante la entrada de la proposición P se ejecute una reacción en cadena que termine liberando como salida la proposición R. Esto es un ejemplo de Modus Ponens Iterado.

De igual manera, ante la entrada de la proposición $\neg R$, se ejecuta una reacción en cadena que termina liberando como salida la proposición $\neg P$. Esto es un ejemplo de Modus Tollens Iterado.

Ambos ejemplos están descritos en la Figura 4.5.

4.4.3. Funcionamiento recursivo

A partir de la estructura básica de codificación de una regla simple representada en la Figura 4.2, podemos utilizarla de manera recursiva para representar reglas compuestas, o lo que es lo mismo, reglas cuyo antecedente o consecuente es otra regla. Para ilustrar el proceso, la Figura 4.6 va a representar la siguiente regla compuesta genérica: $P \rightarrow (Q \rightarrow R)$:

1. Se codifica la negación del antecedente ($\neg P$) en una hebra, haciendo que su segmento $\neg p_1$ hibride con el segmento de subíndice 2 que quedará libre en la codificación del consecuente $Q \rightarrow R$.

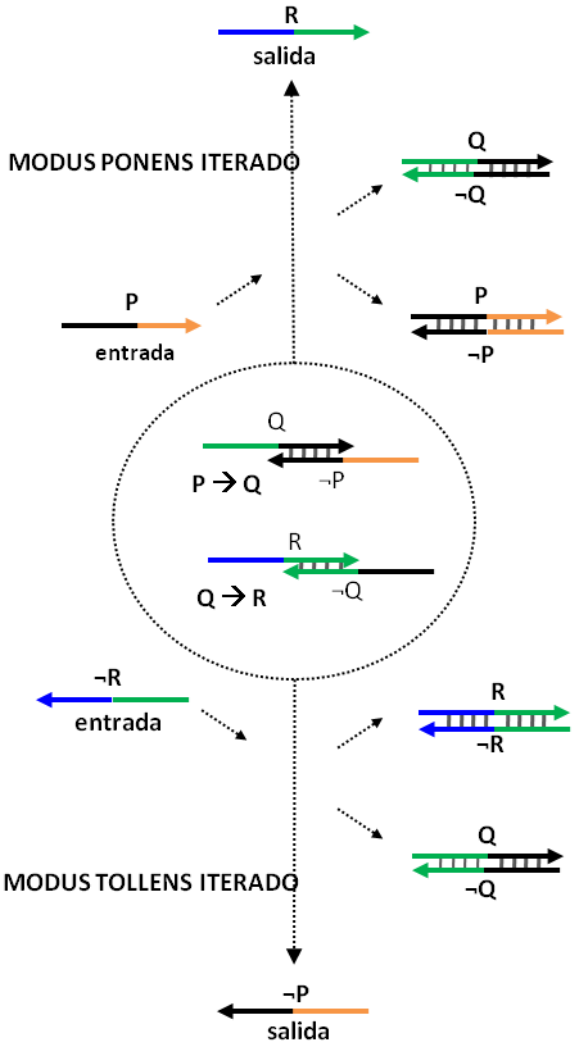

Figura 4.5: Utilización de las reglas de inferencia Modus Ponens y Modus Tollens con el dispositivo 1 en funcionamiento iterativo.

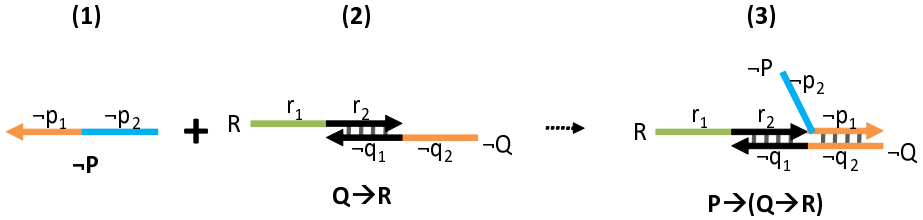

Figura 4.6: Realización del dispositivo 1 en funcionamiento recursivo.

2. Se codifica la regla simple $Q \rightarrow R$ como se ha visto en la Figura 4.2. Dado que es el consecuente de una regla compuesta, debe hibridar un segmento de índice 2 con la hebra que codifique al antecedente; en este caso el segmento $\neg q_2$.

3. La estructura final del dispositivo está formada por una molécula compuesta por la hibridación de las hebras que representan a $\neg P$, $\neg Q$ y R en sus segmentos complementarios.

Una vez codificada la regla compuesta $P \rightarrow (Q \rightarrow R)$, al igual que en el funcionamiento básico, ante la llegada de una hebra codificando la proposición P, se ejecuta la regla de inferencia Modus Ponens y se libera como salida la regla $Q \rightarrow R$.

Dado que cualquier implicación $A \rightarrow B$ puede reescribirse como la disyunción $\neg A \vee B$, podemos de igual manera reescribir la regla compuesta $P \rightarrow (Q \rightarrow R)$ como $\neg P \vee (\neg Q \vee R)$. Esta última expresión es equivalente a $(\neg P \vee \neg Q) \vee R$ (por la propiedad asociativa de la disyunción), que a su vez es equivalente a $\neg(P \wedge Q) \vee R$ (por las Leyes de Morgan). Esta última expresión puede reescribirse de nuevo como implicación: $P \wedge Q \rightarrow R$.

El interés principal de este modo de funcionamiento se justifica en que abre la posibilidad de representar reglas en las que el antecedente está formado por una conjunción de proposiciones, tal y como se ha mostrado en el párrafo anterior.

La codificación de reglas compuestas cuyo antecedente es una regla, como por ejemplo $(P \rightarrow Q) \rightarrow R$, se realizaría de manera análoga y simétrica a lo descrito en esta sección y en la Figura 4.6.

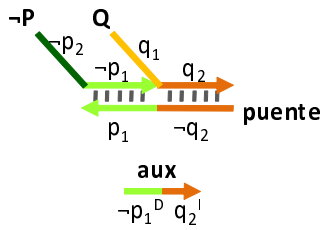

Figura 4.7: Estructura del dispositivo 2 para representación de reglas lógicas

4.5. Dispositivo 2: inferencia con hibridación competitiva en dos pasos

El segundo dispositivo presentado en este capítulo comprende cuatro hebras de ácidos nucleicos, definidas de la siguiente manera (ver Figura 4.7):

- Hebra de ADN que codifica la negación de la proposición que es antecedente de la regla, de acuerdo con el modelo de representación de proposiciones.
- Hebra de ADN que codifica la proposición que es consecuente de la regla, de acuerdo con el modelo de representación de proposiciones.
- Hebra "puente" de ADN, que comprende una secuencia de nucleótidos dividida en dos segmentos con subíndices 1 y 2, caracterizada porque su segmento de índice 1 es complementario al segmento de índice 1 de la hebra que codifica al antecedente de la regla, y su segmento de índice 2 es complementario al segmento de índice 2 de la hebra que codifica al consecuente de la regla.
- Hebra auxiliar de ADN, que comprende una secuencia de nucleótidos caracterizada por ser totalmente complementaria a la secuencia de nucleótidos de la hebra puente, excepto en un prefijo y un sufijo de longitud prefijada en dicha secuencia de la hebra puente.

El objetivo del dispositivo 2 es la representación de una regla lógica simple, que genéricamente podemos denotar como $P \rightarrow Q$, y queda relacionada con el dispositivo de la siguiente manera: la hebra cuyo segmento de subíndice 1 está hibridado con la hebra puente, codifica a la negación del antecedente ($\neg P$); la hebra cuyo segmento de índice 2 está hibridado con la hebra puente, codifica al consecuente (Q).

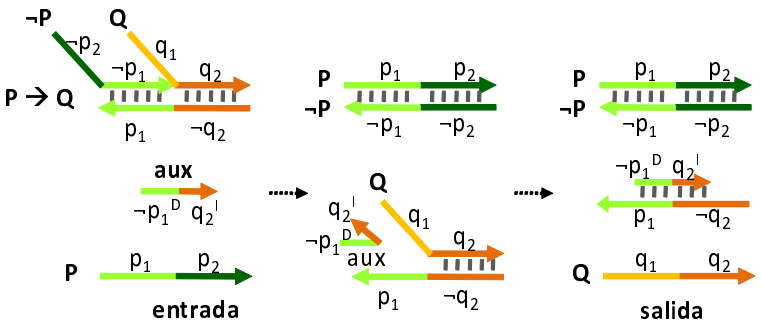

Figura 4.8: Utilización de la regla de inferencia Modus Ponens con el dispositivo 2 en su funcionamiento básico.

4.5.1. Funcionamiento básico

Un dispositivo en su estado inicial, que codifique la regla simple $P \rightarrow Q$ según la estructura descrita en la Figura 4.7, variará su configuración en presencia de dos posibles hebras de entrada: ante la aparición del antecedente P, el dispositivo funcionará de acuerdo a la regla de inferencia Modus Ponens, liberando como salida la hebra que codifica a Q. Por el contrario, ante la aparición de la negación del consecuente, $\neg Q$, el dispositivo funcionará de acuerdo a la regla de inferencia Modus Tollens, liberando como salida la hebra que codifica a $\neg P$.

- *Modus Ponens*. El proceso está descrito en la Figura 4.8:

  1. Ante la entrada de una hebra codificando la proposición P, esta hibrida en su mitad p_2 con el segmento $\neg p_2$ del dispositivo. Posteriormente, el segmento p_1 de la entrada P hibrida con el segmento $\neg p_1$ de la hebra $\neg P$ del dispositivo, completando el primer paso de hibridación competitiva: se forma una hebra doble totalmente complementaria con P y $\neg P$, que se libera del dispositivo.
  2. La hebra puente tiene ahora su segmento p_1 libre, pudiendo hibridar con la hebra auxiliar en su segmento $\neg p_1^D$; posteriormente, el segmento q_2^I de la hebra auxiliar se hibrida completamente con la hebra puente, liberando la salida del dispositivo: la hebra que codifica a Q.

- *Modus Tollens*. El proceso está descrito en la Figura 4.9:

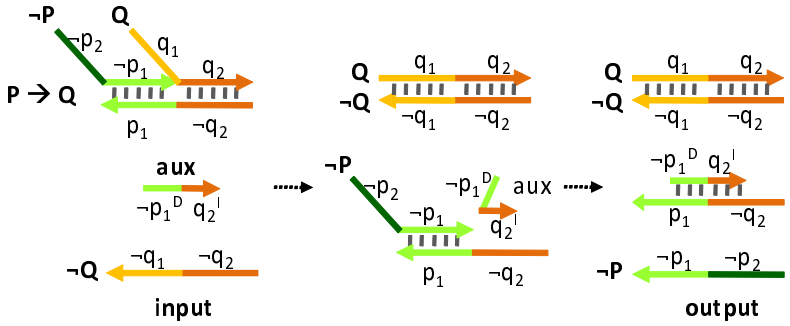

Figura 4.9: Utilización de la regla de inferencia Modus Tollens con el dispositivo 2 en su funcionamiento básico.

1. Ante la entrada de una hebra codificando la proposición $\neg Q$, esta hibrida en su mitad $\neg q_1$ con el segmento q_1 del dispositivo. Posteriormente, el segmento $\neg q_2$ de la entrada $\neg Q$ hibrida con el segmento q_2 de la hebra Q del dispositivo, completando el primer paso de hibridación competitiva: se forma una hebra doble totalmente complementaria con Q y $\neg Q$, que se libera del dispositivo.
2. La hebra puente tiene ahora su segmento $\neg q_2$ libre, pudiendo hibridar con la hebra auxiliar en su segmento q_2^I; posteriormente, el segmento p_1^D de la hebra auxiliar se hibrida completamente con la hebra puente, liberando la salida del dispositivo: la hebra que codifica a $\neg P$.

4.5.2. Funcionamiento iterativo

Dado que el dispositivo utiliza como entrada y salida hebras simples de la misma longitud, es posible disponer varios de ellos formando cascadas de inferencia. De esta manera, podemos tener dos reglas simples, $P \rightarrow Q$ y $Q \rightarrow R$, de forma que ante la entrada de la proposición P se ejecute una reacción en cadena que termine liberando como salida la proposición R. Esto es un ejemplo de *Modus Ponens Iterado*.

De igual manera, ante la entrada de la proposición $\neg R$, se ejecuta una reacción en cadena que termina liberando como salida la proposición $\neg P$. Esto es un ejemplo de *Modus Tollens Iterado*.

Ambos ejemplos están descritos en la Figura 4.10. La repetición o iteración de estas operaciones no se limita a dos casos sucesivos. También es posible encadenar un número

de reglas mayor.

4.5.3. Funcionamiento recursivo

A partir de la estructura básica de codificación de una regla simple, podemos utilizarla de manera recursiva para representar reglas compuestas, o lo que es lo mismo, reglas cuyo antecedente o consecuente es otra regla. Para ilustrar el proceso, la Figura 4.11 va a representar la siguiente regla compuesta genérica: $P \rightarrow (Q \rightarrow R)$:

1. Se codifica la negación del antecedente ($\neg P$) en una hebra, haciendo que su segmento $\neg p_1$ hibride con el segmento p_1 de su correspondiente hebra puente. El segmento de índice 2 de la hebra puente se diseña de manera que hibride con el segmento de índice 2 que quede libre en la codificación del consecuente $Q \rightarrow R$.

2. Se codifica la regla simple $Q \rightarrow R$ como se ha visto en las secciones anteriores. Dado que es el consecuente de una regla compuesta, debe hibridar un segmento de índice 2 con la hebra puente de la regla compuesta; en este caso el segmento $\neg q_2$.

3. La estructura final del dispositivo está formada por tres moléculas, de las cuales dos son hebras auxiliares, y la tercera es la agregación de las proposiciones $\neg P$, $\neg Q$ y R con las dos hebras puente descritas en los pasos 1 y 2.

Una vez codificada la regla compuesta $P \rightarrow (Q \rightarrow R)$, al igual que en el funcionamiento básico, ante la llegada de una hebra codificando la proposición P, se ejecuta la regla de inferencia Modus Ponens y se libera como salida la regla $Q \rightarrow R$.

Dado que cualquier implicación $A \rightarrow B$ puede reescribirse como la disyunción $\neg A \vee B$, podemos de igual manera reescribir la regla compuesta $P \rightarrow (Q \rightarrow R)$ como $\neg P \vee (\neg Q \vee R)$. Esta última expresión es equivalente a $(\neg P \vee \neg Q) \vee R$ (por la propiedad asociativa de la disyunción), que a su vez es equivalente a $\neg(P \wedge Q) \vee R$ (por las Leyes de Morgan). Esta última expresión puede reescribirse de nuevo como implicación: $P \wedge Q \rightarrow R$.

El interés principal de este modo de funcionamiento se justifica en que abre la posibilidad de representar reglas en las que el antecedente está formado por una conjunción de proposiciones, tal y como se ha mostrado en el párrafo anterior.

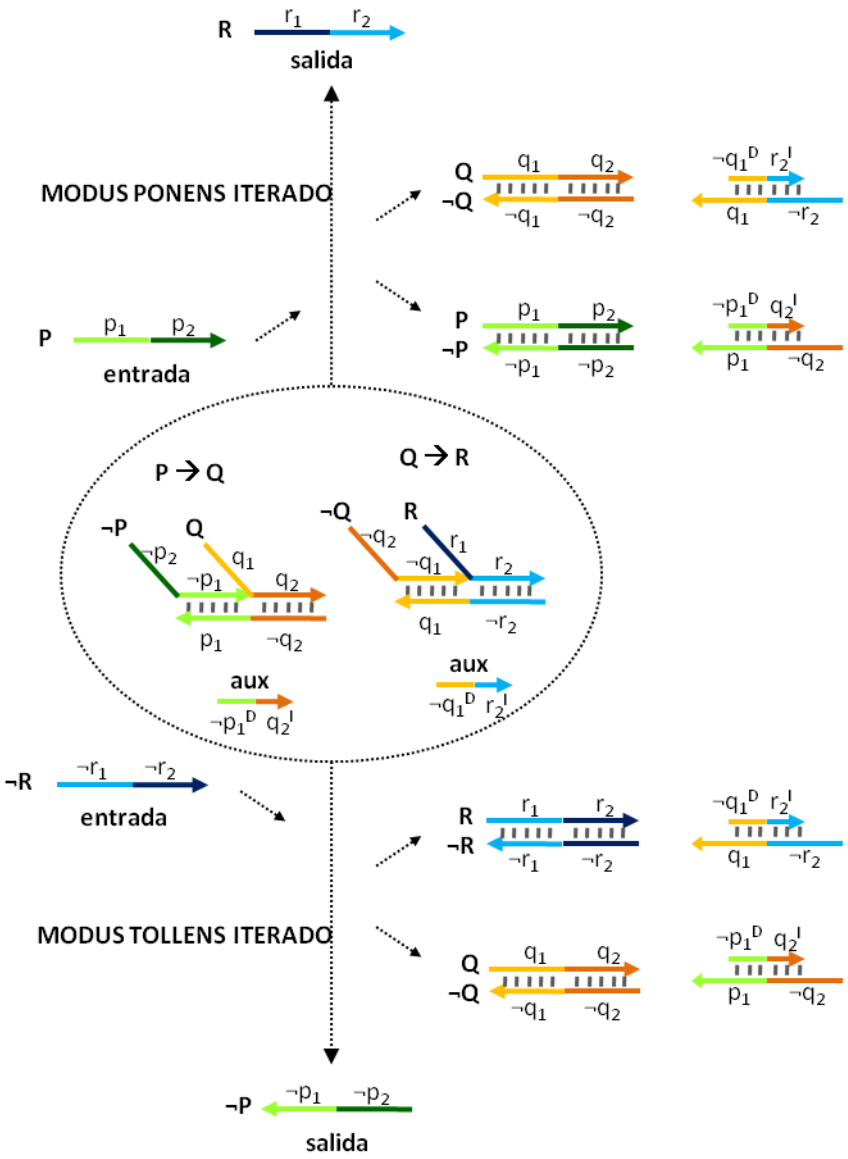

Figura 4.10: Utilización de las reglas de inferencia Modus Ponens y Modus Tollens con el dispositivo 2 en funcionamiento iterativo.

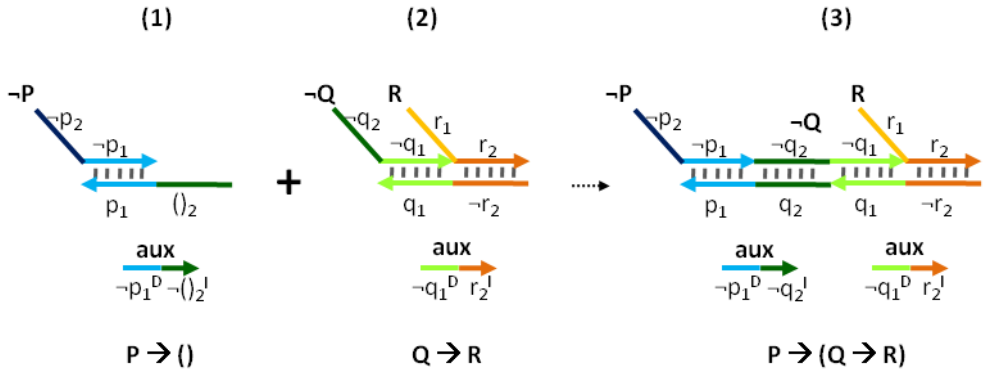

Figura 4.11: Realización del dispositivo 2 en funcionamiento recursivo.

La codificación de reglas compuestas cuyo antecedente es una regla, como por ejemplo $(P \rightarrow Q) \rightarrow R$, se realizaría de manera análoga y simétrica a lo descrito en esta sección y en la Figura 4.11.

4.6. Dispositivo 3: representación de cláusulas lógicas

El tercer dispositivo presentado en este capítulo comprende un número variable de hebras de ácidos nucleicos, definidas de la siguiente manera:

- Un conjunto de hebras, cada una de las cuales representan a una proposición, de acuerdo con el modelo de representación de proposiciones.
- Una hebra puente que comprende una secuencia de nucleótidos caracterizada por estar dividida en tantos segmentos como el número de proposiciones representadas en el dispositivo. Cada uno de estos segmentos es totalmente complementario al segmento con índice 1 de la hebra que codifica a la proposición asociada a dicho segmento de la hebra puente.
- Cuando el dispositivo se utiliza para evaluar asignaciones de verdad a fórmulas lógicas, utiliza también una hebra auxiliar, que comprende una secuencia de nucleótidos totalmente complementaria a la hebra puente.

Este dispositivo tiene por objeto representar y codificar una cláusula lógica. Al igual que las reglas de los dispositivos 1 y 2, las cláusulas lógicas se construyen a partir de

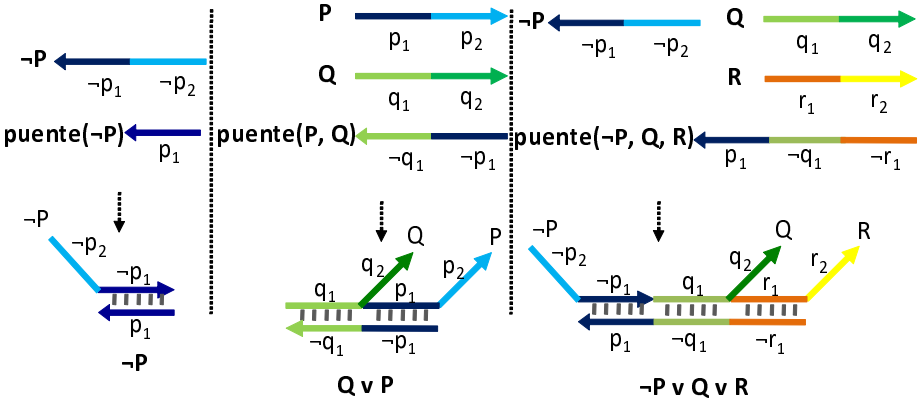

Figura 4.12: Ejemplos de codificación de cláusulas utilizando el dispositivo 3.

proposiciones, las cuales se siguen codificando de acuerdo a lo descrito en la Figura 4.1. La Figura 4.12 muestra 3 ejemplos de codificación de cláusulas mediante el dispositivo 3.

Respecto a la construcción de la cláusula, veremos que es muy similar a la del dispositivo 2, pero no se restringe a dos proposiciones sino que puede contener cualquier número de las mismas. La base del dispositivo va a ser también una hebra puente, pero en este caso tendrá un segmento por cada proposición contenida en la cláusula. Dicho segmento se hibrida con el segmento de la correspondiente proposición que tenga subíndice 1, dejando todos los segmentos de las proposiciones con subíndice 2 sin hibridar. La Figura 4.12 muestra el ejemplo de cómo representar las cláusulas $\neg P$, $Q \vee P$ y $\neg P \vee Q \vee R$).

4.6.1. Resolución

Recordemos el principio de resolución: Si C y C' son cláusulas y P es una proposición, entonces cualquier asignación de valores que satisface $C \vee P$ y $C' \vee \neg P$, satisface también $C \vee C$'.

La Figura 4.13 muestra como este principio puede representarse en ADN con el dispositivo propuesto: las cláusulas iniciales C_1 y C_2 interaccionan entre sí, formando una molécula intermedia donde los segmentos a_2 y $\neg a_2$ quedan hibridados. Con el paso del tiempo, el sistema tiende a evolucionar a una configuración más estable, buscando minimizar la energía libre de Gibbs. Así pues, las hebras A y $\neg A$ terminarán formando una

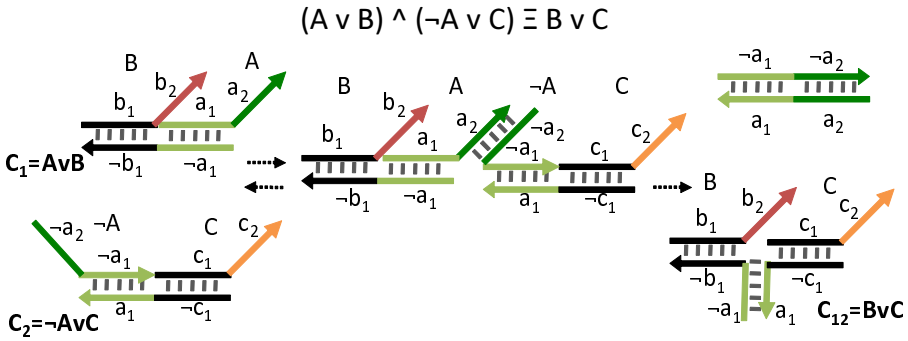

Figura 4.13: Paso básico de la regla de resolución, utilizada por el dispositivo 3.

hebra doble. Paralelamente, las hebras puente de C_1 y C_2 hibridarán en sus segmentos $\neg a_1$ y a_1, resultando en la molécula que codifica la cláusula final $C_{12} = B \vee C$.

Repasemos ahora la definición de *silogismo hipotético*: Si tenemos las reglas simples $P \to Q$ y $Q \to R$, entonces podemos deducir la regla $P \to R$.

El ejemplo de la Figura 4.13 ha sido elegido estratégicamente para que también ilustre este tipo de silogismo. Dado que una implicación lógica $P \to Q$ puede reescribirse como la disyunción $\neg P \vee Q$, podemos realizar este ejercicio con las cláusulas de la figura, obteniendo las siguientes expresiones: $C_1 = \neg B \to A$, $C_2 = A \to C$, $C_{12} = \neg B \to C$. Puede verse con facilidad que las tres cláusulas son consistentes con la definición de silogismo hipotético. No es casualidad, ya que el silogismo puede definirse a nivel teórico como un caso particular de resolución, donde las cláusulas están formadas por dos proposiciones.

4.6.2. Evaluación de fórmulas

Dada una fórmula lógica a la que se ha asociado una asignación de valores de verdad de las proposiciones, la evaluación de la fórmula consiste en determinar si dicha asignación de valores de las proposiciones satisface la fórmula.

El dispositivo 3 (ver figura 4.14) puede también utilizarse para este efecto. Para ello son necesarias las siguientes variantes:

- Añadir una hebra auxiliar al dispositivo que sea complementaria a la hebra puente. En uno de sus extremos (3' o 5') debe fijarse una partícula atenuadora de

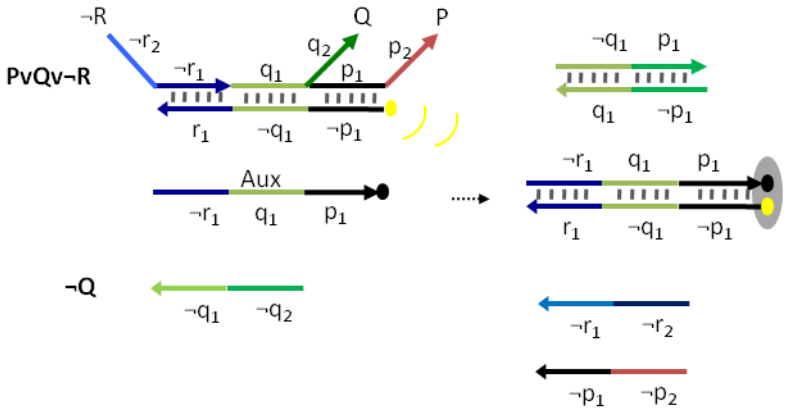

Figura 4.14: Utilización del dispositivo 3 para evaluar una asignación de valores de verdad a una fórmula lógica.

luminosidad (quencher), capaz de absorber la luz emitida por un fluoróforo.

- Fijación un fluoróforo en el extremo de la hebra puente que es antiparalelo al extremo de la hebra auxiliar donde se ha fijado la partícula atenuadora.

Con esto el dispositivo está listo para interactuar con la asignación de valores de verdad. En su configuración inicial, al no estar enfrentados el fluoróforo y la partícula atenuadora, el sistema emite fluorescencia. El siguiente paso es codificar la asignación de valores de verdad, introduciendo en la disolución, por cada proposición, una hebra codificando el valor opuesto al que se pretende asignar a la proposición. Es decir, si queremos evaluar la fórmula para $P = cierto$, debemos introducir en la disolución una hebra que codifique $P = falso$ ($\neg P$).

La Figura 4.14 muestra un ejemplo de asignación de valor de verdad a una proposición que satisface una fórmula. Dicha fórmula está compuesta por una única cláusula, $P \vee Q \vee \neg R$, y para evaluar si $Q = cierto$ la satisface, se introduce en la disolución la hebra que codifica a $Q = falso$ ($\neg Q$). El sistema evoluciona de la siguiente manera:

1. $\neg Q$ se hibrida totalmente con la hebra Q del dispositivo, dejando el segmento $\neg q_1$ de la hebra puente libre.

2. La hebra auxiliar hibrida su segmento q_1 con el segmento $\neg q_1$ de la hebra puente.

3. La hebra auxiliar se hibrida completamente con la hebra puente, desplazando al resto de hebras de proposiciones. Al juntarse el fluoróforo de la hebra puente con la partícula atenuadora de la hebra auxiliar, eliminando la fluorescencia.

Así pues, la eliminación de luminosidad fluorescente implica la que la asignación de verdad satisface la fórmula. Si la fórmula estuviera compuesta por varias cláusulas, la asignación de verdad tendría que ser capaz de "apagar" la luminosidad en todas ellas. Por el contrario, el mantenimiento de la luminosidad implica la asignación de verdad no satisface la fórmula.

4.7. Caso de estudio: diagnóstico *in vitro*

Como ya se ha descrito en el capítulo 2, Benenson et al. (11) desarrollaron en 2004 un autómata de ADN que, tras percibir una serie indicadores de enfermedad en forma de secuencias de ARN mensajero, es capaz de proporcionar un diagnóstico positivo o negativo, liberando el tratamiento o el supresor del tratamiento, respectivamente. Dicho sistema implementaba un autómata molecular que realizaba las transiciones de estado gracias a la enzima de restricción Fok I.

En este caso de estudio se propone la implementación de un sistema equivalente, pero utilizando un paradigma computacional y un proceso biológico distintos: en lugar de una máquina de estados finitos, lógica proposicional; en lugar de transiciones de estado mediante enzimas de restricción, los métodos basados en hibridación competitiva implementados por los dispositivos 1 y 2 descritos en este capítulo.

Al igual que en Benenson et al. (11), se utiliza un indicador biológico (por ejemplo, una alta concentración de un ARNm específico) para iniciar el sistema. Una puerta traductora (15) captura dicho indicador biológico y libera una hebra de entrada en el sistema, que podrá interaccionar con las distintas implicaciones (el programa lógico), realizar inferencia (modus ponens o modus tollens) y liberar las proposiciones de salida (diagnóstico). Como paso final, las proposiciones de salida se traducen a una señal externa, que podría ser fluorescencia (FRET) o incluso la liberación de una molécula de tratamiento de la enfermedad.

El siguiente ejemplo muestra más en detalle la aplicación propuesta:

- Síntomas: tres tipos distintos de ARN mensajero: $ARNm_1$, $ARNm_2$ y $ARNm_3$.

- Diagnósticos: dos tipos distintos de enfermedad: enf_A y enf_B.
- Una alta concentración de $ARNm_1$ determina la existencia de enf_A. Esto significa que, siempre que enf_A esté presente, existirá una alta concentración de $ARNm_1$ como síntoma. Y viceversa, una alta concentración de $ARNm_1$ siempre determina la existencia de enf_A. Si lo expresamos mediante lógica proposicional, tenemos las siguientes expresiones:
  - $enf_A \rightarrow ARNm_1$
  - $ARNm_1 \rightarrow enf_A$
- Una baja concentración de $ARNm_2$ es indicadora de enf_B. Esto significa que, siempre que enf_B esté presente, existirá una baja concentración de $ARNm_2$. Sin embargo, una baja concentración de $ARNm_2$ no siempre implica la existencia de enf_B. Si lo expresamos mediante lógica proposicional, tenemos la siguiente expresión;
  - $enf_B \rightarrow \neg ARNm_2$

Imaginemos que el sistema recibe como entrada altas concentraciones de $ARNm_1$ y $ARNm_2$.

- la proposición de entrada $ARNm_1$ coincide con el antecedente de la implicación $ARNm_1 \rightarrow enf_A$. Por tanto, se activa la regla de inferencia modus ponens y se libera como salida la proposición enf_A.
- La proposición $ARNm_2$ coincide con la negación del consecuente de la implicación $enf_B \rightarrow \neg ARNm_2$. Por consiguiente, se activa la regla de inferencia modus tollens y se libera como salida la proposición $\neg enf_B$.

Cada proposición de salida se diseña de manera que active un color de fluorescencia específico, de manera que la salida del sistema pueda ser fácilmente reconocible y medible.

Capítulo 5

Análisis de los resultados

Los modelos de dispositivos de inferencia con ADN presentados en este capítulo son al menos tan expresivos como el sistema desarrollado en Ran et al. (59). Aunque este último presenta mejor velocidad de reacción (gracias a la utilización de enzimas), nuestros modelos presentan dos características muy interesantes: representación explícita de la negación lógica (63), y ausencia de enzimas de restricción o cualquier otro tipo de reactivo distinto de los ácidos nucleicos.

La capacidad para representar de manera explícita la negación lógica incrementa la potencia de las reglas del sistema. Gracias a esta característica, la misma implicación $P \rightarrow Q$ puede activarse en presencia de su antecedente (mediante la aplicación de la regla de inferencia modus ponens) o en presencia de la negación lógica de su consecuente (mediante la aplicación de la regla de inferencia modus tollens). Esto también implica que los dispositivos puedan ofrecer una proposición negada como salida. En lo que respecta al dispositivo 3, esta codificación dual de los valores de verdad de las proposiciones es la clave para que se inicie el paso de resolución.

La representación dual de proposiciones afirmadas y negadas también proporciona un mecanismo implícito de corrección de errores. Imaginemos que la salida esperada del sistema es Q, pero la disolución contiene hebras de $\neg Q$ disociadas espontáneamente (por error) de otras reglas. Las hebras erróneas hibridarían con las hebras que representan la salida esperada, formando hebras dobles neutras (con extremos romos) que no podrían interaccionar con ningún otro elemento del sistema. Como la concentración de

las hebras correctas sería mayor que las erróneas, la señal correcta se propagaría libre de error, aunque algo atenuada.

Un artículo publicado por Chiniforooshan et al. (64) introdujo una lista de propiedades que son deseables en los diseños de dispositivos de computación biomolecular: *escalabilidad*, *sensibilidad al tiempo*, empheficiencia energética y *digitalidad*. A continuación se muestra en qué grado los dispositivos presentados en esta tesis cumplen dichas propiedades:

Escalabilidad *La preparación de las diferentes hebras de ADN puede realizarse en un único tubo de ensayo, sin necesidad de utilizar un tubo por cada tipo de hebra.* En nuestro caso se cumple parcialmente: las hebras que codifican proposiciones pueden dividirse en dos grupos, positivas y negativas, de manera que cada grupo puede prepararse de manera conjunta en el mismo tubo de ensayo. Sin embargo, has hebras de ADN "puente" y "auxiliar" de los dispositivos 2 y 3 deben ser preparadas de forma separada, ya que pueden contener partes mutuamente complementarias.

Sensibilidad al tiempo *Tras la realización de cómputos iniciales, el sistema todavía es capaz de reaccionar a cambios en las entradas y procesar una nueva salida.* Mientras existan en el sistema dispositivos que no han sido consumidos tras los cómputos iniciales, el sistema seguirá respondiendo ante entradas posteriores. Dejará de responder en el momento en que todos los dispositivos hayan sido consumidos procesando entradas.

Eficiencia energética *Ante entradas al sistema en condiciones ideales (por ejemplo, sin contaminación por hebras no deseadas), el sistema llega a alcanzar un estado estable donde la salida también presenta condiciones ideales y el sistema permanece en equilibro energético.* Se cumple en todos los modelos presentados.

Digitalidad *Las reacciones de ADN que implementa el dispositivo contemplan la restauración de señal, cuando la concentración de una hebra de ADN específica que codifica un valor lógico es bajo, logrando así una abstracción digital sobre los valores de concentración analógicos.* Ninguno de los modelos presentados cumple con

esta propiedad, por lo que se asume que en los casos en que fuera necesario, se podrían utilizar amplificadores como los presentados en trabajos anteriores (15). No obstante, la mejora de esta propiedad en los dispositivos será uno de los objetivos futuros tras la finalización de la presente tesis.

La ausencia de enzimas de restricción permite modelos más sencillos, más fácilmente reaplicables en distintas condiciones de temperatura y pH, y más cercanos a ser utilizados en potenciales aplicaciones *in vivo*.

A continuación se ofrece un análisis de las bondades y limitaciones de cada uno de los modelos presentados en este capítulo.

5.1. Inferencia con el dispositivo 1

La principal ventaja que ofrece este modelo es su simplicidad. Como puede verse en la figura 4.2, para construir las reglas de implicación, basta con reservar regiones con secuencias complementarias en las hebras que codifican al antecedente y al consecuente, de modo que puedan mantenerse unidas en ausencia de otras hebras de entrada. Si bien el antecedente de la regla o la negación lógica del consecuente están presentes, la correspondiente regla de inferencia, modus ponens o modus tollens, se aplica mediante un único proceso de hibridación competitiva. Este modelo de dispositivo es una buena elección siempre y cuando el sistema que se pretende implementar pueda adaptarse a las siguientes restricciones:

- No es posible representar ciclos. Esto ocurre como consecuencia de tener que construir los dispositivos mezclando "datos" y "programa" en las mismas hebras. Tras haber diseñando las implicaciones $P \to Q$ y $Q \to R$, no hay posibilidad de representar $R \to P$. Lógicamente, la doble implicación $R \leftrightarrow P$ tampoco puede ser representada.

- Cuando se encadenan tres implicaciones consecutivas, puede haber problemas de "interferencia de señal". Si imaginamos un programa lógico con las reglas de implicación (a) $P \to Q$, (b) $Q \to R$ y (c) $R \to S$, el extremo cohesivo de Q en (a) es complementario al extremo cohesivo de $\neg R$ en (c). Si la proposición R fuera a

ser una entrada del sistema, su velocidad de reacción con la regla de implicación (c) se vería reducida por su competencia con el extremo cohesivo de (a).

5.2. Inferencia con el dispositivo 2

Si el programa lógico que se pretende construir está afectado por alguna de las restricciones del modelo anterior, estas pueden solventarse utilizando el dispositivo 2. En este modelo, las secuencias de bases de cada una de las proposiciones son diferentes, incluso si forman parte de la misma regla de implicación. Esto se consigue diseñando las implicaciones de forma que "datos" y "programa" estén separados: dadas dos proposiciones P y Q, siempre va a ser posible construir una secuencia de ADN que hibride con el segmento 1 de $\neg P$ y el segmento 2 de Q (ver figura 4.7). El inconveniente del dispositivo 2 respecto al dispositivo 1 es su menor velocidad. Independientemente de la regla de inferencia que se vaya a aplicar (modus ponens o modus tollens), el dispositivo 2 siempre necesitará dos procesos de hibridación competitiva para liberar completamente la proposición de salida, reduciendo así su velocidad en un factor de 2 respecto al dispositivo 1.

La aplicación más directa que se puede pensar, tanto para el dispositivo 1 como el 2, es su utilización como puertas lógicas bidireccionales ("hacia adelante" aplicando modus ponens, "hacia atrás" aplicando modus tollens) en la implementación de biosensores de ácidos nucleicos. Desde una visión más computacional (65), las reglas de implicación implementadas por estos dispositivos se pueden interpretar como registros de una base de datos *in vitro* que codifican información biológica. Dicha base de datos no necesitaría un soporte digital, así que podría procesar directamente como entrada muestras biológicas procedentes de ADN o ARN procedentes de diversos organismos, implementar búsquedas booleanas para dichas entradas, y generar las distintas salidas también con un sustrato biológico.

5.3. Inferencia con el dispositivo 3

El modelo que implementa este dispositivo permite la representación de fórmulas lógicas en *forma normal conjuntiva* (FNC) y ejecuta de manera autónoma un mecanismo de resolución que actúa en cualquier pareja de cláusulas que presenten alguna

proposición complementaria.

Una de las posibles aplicaciones de este modelo es la simplificación de fórmulas. A modo ilustrativo, podemos analizar el silogismo hipotético (caso particular de resolución entre cláusulas de dos proposiciones) como optimizador de reglas. Imaginemos una base de datos hecha con ADN, utilizando reglas codificadas siguiendo el modelo del dispositivo 3. Al añadir nuevas reglas a esa base de datos que mantengan alguna relación de transitividad con las reglas originales de la base de datos (o entre las mismas reglas de entrada), se aplicará entre ellas el silogismo hipotético, logrando simplificar el número de reglas de la base de datos. Tras dejar pasar un tiempo suficiente, bajo las condiciones de temperatura y pH adecuadas, las nuevas reglas podrán identificarse por su mayor longitud respecto a las reglas originales (aplicando la técnica de electroforesis en gel), para posteriormente separarlas, replicarlas mediante PCR y añadirlas a la base de datos original. Como resultado de este proceso la base de datos de ADN no incrementará su conocimiento, pero los cómputos afectados reducirán su tiempo de ejecución gracias al menor número de reacciones correspondientes a reglas intermedias.

Mediante la aplicación sucesiva de diversos pasos de resolución, este modelo también es capaz de aplicar la técnica demostración de teoremas llamada *refutación por resolución*. El objetivo de dicha técnica es demostrar si una fórmula w concreta en FNC (la entrada del sistema) se puede demostrar a partir de otro conjunto de fórmulas Δ en FNC (el programa lógico). Para ello, se añade la negación de la fórmula de entrada, $\neg w$, al conjunto de fórmulas Δ, y se realizan sucesivas iteraciones de resolución entre todas las cláusulas. Si al final del proceso se logra derivar la cláusula vacía, se concluye que w es demostrable mediante Δ ya que su negación deriva en una contradicción. Visto con este enfoque, los dispositivos de este modelo podrían integrarse como parte de un nanoautómata autónomo, capaz de ejecutar un programa lógico como respuesta a diferentes entradas. El punto pendiente de esta aplicación sería el paso final de detección de la cláusula vacía. La solución más directa es la utilización de electroforesis en gel para identificarla por longitud. Pero siendo la autonomía una de las características más deseables en este tipo de nanoautómatas, sería necesario un proceso autónomo de detección, quizás basado en técnicas FRET, evitando así el uso de la electroforesis.

El siguiente paso será abordar la resolución del *Problema de Satisfacibilidad de Fórmulas Booleanas (SAT)*. Siendo capaces de aplicar múltiples refutaciones en paralelo buscando derivar la cláusula vacía, su resolución es posible desde un punto de vista teórico. Si al final del proceso no se identifica ninguna molécula como potencial candidata a ser la cláusula vacía, se podría concluir SAT como solución al problema. La resolución del problema SAT es otra de las investigaciones que se han abierto en esta tesis y se seguirán desarrollando en el futuro.

La utilización de este dispositivo para la evaluación de fórmulas tiene aplicación directa en la implementación se biosensores. Si se observa el funcionamiento del dispositivo descrito en la figura 4.14, se puede interpretar que implementa una puerta NAND: la salida sólo será "falso" en caso de que ninguna de las entradas tomen valor "falso" (entendiendo que una entrada toma valor "falso" cuando es complementaria (y por tanto codifica el valor opuesto) a una de las proposiciones que incluye el dispositivo.

Capítulo 6

Conclusiones y futuras líneas de investigación

Esta tesis presenta tres modelos sencillos para realizar computación lógica aprovechando el fenómeno de hibridación competitiva del ADN, que se engloban en la llamada *nanotecnología estructural con ADN*. Todos los modelos son autónomos, excepto en los casos en que la salida no puede medirse utilizando técnicas FRET y es necesario utilizar electroforesis en gel. Ninguno de los modelos utiliza enzimas y su implementación puede basarse en los modelos ya derivados experimentalmente (17, 18). De acuerdo con las propiedades examinadas en Chiniforooshan et al. (64), podemos afirmar que los tres modelos son parcialmente escalables, sensibles al tiempo y energéticamente eficientes.

En diversos trabajos anteriores (15, 59), el valor de verdad "cierto" se representa con una hebra de ADN, mientras que el valor de verdad "falso" se representaba con la ausencia de dicha hebra. Los modelos desarrollados en el presente trabajo codifican el valor de verdad "falso" con la hebra de ADN complementaria a la utilizada para codificar el valor "cierto" (63). La utilidad de esta alternativa de codificación es doble:

- Para el caso de los dispositivos 1 y 2, posibilita obtener proposiciones negadas como salidas válidas, permitiendo que la misma implicación de pueda realizar inferencia "hacia adelante" (modus ponens) o "hacia atrás" (modus tollens);
- En lo que respecta al dispositivo 3, esta codificación dual de los valores de verdad de las proposiciones es la clave para que se inicie el paso de resolución.

- Además, introduce un mecanismo implícito de cancelación de errores, que incrementa la escalabilidad del sistema permitiendo más pasos de inferencia en cascada con una relación señal-ruido limitada y más controlada.

Los dispositivos 1 y 2 permiten implementar reglas de implicación sencillas (como por ejemplo $P \rightarrow Q$) y aplicar de manera autónoma las reglas de inferencia *modus ponens* y *modus tollens*. Pueden tener utilidad en la construcción de sensores bidireccionales de ácidos nucleicos, así como en el mantenimiento de bases de datos con registros en sustrato biológico.

El dispositivo 3 permite representar cláusulas lógicas en forma normal conjuntiva (FNC) y aplicar entre ellas las reglas de inferencia de *resolución* y *silogismo hipotético* de manera autónoma. El modelo tiene evidente utilidad en la simplificación de fórmulas en FNC, pero para llegar a su máximo potencial, necesitará ser capaz de implementar de manera totalmente autónoma (con un método distinto de la electroforesis en gel) la técnica de demostración de teoremas por refutación. Tras alcanzar este hito, los nanoautómatas implementados a partir del modelo resultante tendrían un gran potencial de aplicación en la mejora de técnicas de análisis genómico. Este tema sigue abierto tras la finalización de la presente tesis, y seguirá siendo desarrollado en el futuro. La utilización del dispositivo 3 para la evaluación de fórmulas en FNC también tiene aplicación en la construcción de sensores bidireccionales de ácidos nucleicos y en el mantenimiento de bases de datos con registros en sustrato biológico.

Además de ADN, todos los modelos pueden aplicarse a cualquier otro tipo de ácido nucleico (como ARN de interferencia o microRNA). Todos ellos pueden habilitar aplicaciones muy interesantes en en área de los circuitos de ARN sintético, como por ejemplo, modular la expresión de diversos genes a partir de la interacción de los dispositivos presentados y el ARN post-transcripcional. Este tipo de aplicaciones tienen actualmente una gran relevancia por su extraordinaria potencialidad en biomedicina, como evidencia la reciente publicación en la revista *Science* de un autómata biomolecular de diagnóstico, capaz de tratar las células cancerígenas sin afectar a las células sanas (61).

El trabajo desarrollado para esta tesis ha dado lugar a la siguiente lista de publicaciones:

- *Dispositivos de ácidos nucleicos para la realización de inferencia lógica.* Solicitud de patente presentada en la Oficina Española de Patentes y Marcas (OEPM). Pendiente de resolución.
- *Inference with DNA molecules.* Póster. Unconventional Computation, Lecture Notes in Computer Science. Springer. 2010 (21).
- *Smart bidirectional sensors made of nucleic acid molecules.* Póster. International Conference on Synthetic Biology. Diciembre de 2010 (62).
- *Biomolecular Computers.* Current Bioinformatics. Junio de 2011 (32).

6. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

Bibliografía

[1] Jesús M. Miró and Alfonso Rodríguez-Patón. **Biomolecular Computing Devices in Synthetic Biology**. *International Journal of Nanotechnology and Molecular Computation*, **2**(2):47–64, MarFeb 2010. 1

[2] J. D. Watson and F. H. C. Crick. **Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid**. *Nature*, **171**(4356):737–738, April 1953. 1, 4

[3] Richard J. Roberts. **How restriction enzymes became the workhorses of molecular biology**. *Proceedings of the National Academy of Sciences of the United States of America*, **102**(17):5905–5908, 2005. 1

[4] Kathleen Danna and Daniel Nathans. **Specific cleavage of simian virus 40 DNA by restriction endonuclease of Hemophilus influenzae**. *Proceedings of the National Academy of Sciences*, **68**(12):2913–2917, 1971. 1

[5] John M. S. Bartlett and David Stirling. **A Short History of the Polymerase Chain Reaction**. In John M. Walker, John M. S. Bartlett, and David Stirling, editors, *PCR Protocols*, **226 of *Methods in Molecular Biology*, pages 3–6. Humana Press, New Jersey, August 2003.** 1

[6] L. M. Adleman. **Molecular computation of solutions to combinatorial problems. *Science (New York, N.Y.)*, 266(5187):1021–1024, November 1994. 2, 13, 17, 33**

[7] R. J. Lipton. **DNA solution of hard computational problems. *Science*, 268(5210):542–545, April 1995. 3, 33**

[8] Donald Beaver. Factoring: The DNA solution. In Josef Pieprzyk and Reihanah Safavi-Naini, editors, *Advances in Cryptology Ã¢ÂÂ ASIACRYPT'94*, 917 of *Lecture Notes in Computer Science*, pages 419–423. Springer Berlin / Heidelberg, 1995. 10.1007/BFb0000453. 3, 33

[9] Donald Beaver. Computing with DNA. *Journal of Computational Biology*, 2(1):1–7, 1995. 3, 33

[10] Yaakov Benenson, Tamar Paz-Elizur, Rivka Adar, Ehud Keinan, Zvi Livneh, and Ehud Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414(6862):430–434, November 2001. 3, 19, 29

[11] Yaakov Benenson, Binyamin Gil, Uri Ben-Dor, Rivka Adar, and Ehud Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature*, 429(6990):423–429, May 2004. 3, 19, 29, 53

[12] Rivka Adar, Yaakov Benenson, Gregory Linshiz, Amit Rosner, Naftali Tishby, and Ehud Shapiro. Stochastic computing with biomolecular automata. *Proceedings of the National Academy of Sciences of the United States of America*, 101(27):9960–9965, July 2004. 3, 19, 29

[13] Ehud Shapiro and Yaakov Benenson. Tapping the computing power of biological molecules gives rise to tiny machines that can speak directly to living cells. *Scientific American*, 294(5):44–51, May 2006. 3, 19, 29

[14] B. Yurke, A. J. Turberfield, A. P. Mills, F. C. Simmel, and J. L. Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406(6796):605–608, August 2000. 3, 24, 27, 34

[15] Georg Seelig, David Soloveichik, David Y. Zhang, and Erik Winfree. Enzyme-Free Nucleic Acid Logic Circuits. *Science*, 314(5805):1585–1588, December 2006. 3, 24, 26, 34, 53, 57, 61

[16] David Y. Zhang, Andrew J. Turberfield, Bernard Yurke, and Erik Winfree. Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA. *Science*, 318(5853):1121–1125, November 2007. 3, 24, 34

[17] David Y. Zhang and Erik Winfree. Control of DNA Strand Displacement Kinetics Using Toehold Exchange. *Journal of the American Chemical Society*, 131(47):17303–17314, December 2009. 3, 24, 34, 61

[18] David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, March 2010. 3, 27, 34, 61

[19] Luca Cardelli. Strand Algebras for DNA Computing. In Russell Deaton and Akira Suyama, editors, *DNA Computing and Molecular Programming*, 5877, chapter 2, pages 12–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 3, 27

[20] Andrew Phillips and Luca Cardelli. A programming language for composable DNA circuits. *Journal of The Royal Society Interface*, 6(Suppl 4):S419–S436, August 2009. 3, 27

[21] Alfonso Rodríguez-Patón, José María Larrea, and Iñaki Sainz de Murieta. Inference with DNA Molecules. In Cristian S. Calude, Masami Hagiya, Kenichi Morita, Grzegorz Rozenberg, and Jon Timmis, editors, *Unconventional Computation*, 6079, chapter 25, page 192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 3, 35, 63

[22] Milan N. Stojanovic, Tiffany E. Mitchell, and Darko Stefanovic. Deoxyribozyme-Based Logic Gates. *Journal of the American Chemical Society*, 124(14):3555–3561, April 2002. 3, 21

[23] Milan N. Stojanovic and Darko Stefanovic. A deoxyribozyme-based molecular automaton. *Nature Biotechnology*, 21(9):1069–1074, August 2003. 3, 22

[24] Renjun Pei, Elizabeth Matamoros, Manhong Liu, Darko Stefanovic, and Milan N. Stojanovic. Training a molecular automaton to play a game. *Nature Nanotechnology*, 5(11):773–777, November 2010. 3, 22

[25] Milan N. Stojanovic, Stanka Semova, Dmitry Kolpashchikov, Joanne Macdonald, Clint Morgan, and Darko Stefanovic. Deoxyribozyme-Based Ligase Logic Gates and Their Initial Circuits. *Journal of the American Chemical Society*, 127(19):6914–6915, May 2005. 3, 23

[26] Renjun Pei, Steven K. Taylor, Darko Stefanovic, Sergei Rudchenko, Tiffany E. Mitchell, and Milan N. Stojanovic. **Behavior of Polycatalytic Assemblies in a Substrate-Displaying Matrix**. *Journal of the American Chemical Society*, 128(39):12693–12699, October 2006. 3, 23

[27] Johann Elbaz, Oleg Lioubashevski, Fuan Wang, Francoise Remacle, Raphael D. Levine, and Itamar Willner. **DNA computing circuits using libraries of DNAzyme subunits**. *Nature Nanotechnology*, 5(6):417–422, May 2010. 3, 23

[28] Keller Rinaudo, Leonidas Bleris, Rohan Maddamsetti, Sairam Subramanian, Ron Weiss, and Yaakov Benenson. **A universal RNAi-based logic evaluator that operates in mammalian cells**. *Nature Biotechnology*, 25(7):795–801, May 2007. 4, 31

[29] Stephanie J. Culler, Kevin G. Hoff, and Christina D. Smolke. **Reprogramming Cellular Behavior with RNA Controllers Responsive to Endogenous Proteins**. *Science*, 330(6008):1251–1255, November 2010. 4, 30

[30] Maung N. Win and Christina D. Smolke. **Higher-Order Cellular Information Processing with Synthetic RNA Devices**. *Science*, 322(5900):456–460, October 2008. 4, 30

[31] Jesús Miró Bueno and Alfonso Rodríguez-Patón. **A New Model of Synthetic Genetic Oscillator Based on Trans-Acting Repressor Ribozyme**. 5518:1170–1177, 2009. 4, 30

[32] Iñaki Sainz de Murieta, Jesús Miró-Bueno, and Alfonso Rodríguez-Patón. **Biomolecular Computers**. *Current Bioinformatics*, pages 173–184, June 2011. 13, 14, 19, 20, 22, 25, 26, 63

[33] Masami Hagiya, Masanori Arita, Daisuke Kiga, Kensaku Sakamoto, and Shigeyuki Yokoyama. **Towards Parallel Evaluation and Learning of Boolean μ-Formulas with Molecules**. 48, pages 105–114, 1997. 15, 29

[34] Alfonso Rodríguez-Patón. ***Variantes de la concatenación en computación con ADN***. PhD thesis, July 1999. 16

[35] Kensaku Sakamoto, Daisuke Kiga, Ken Komiya, Hidetaka Gouzu, Shigeyuki Yokoyama, Shuji Ikeda, Hiroshi Sugiyama, and Masami Hagiya.

% bf State transitions by molecules. *Biosystems*, 52(1-3):81 – 91, 1999. 17

[36] Robert Berger. *Undecidability of the Domino Problem (Memoirs ; No 1/66)*. Amer Mathematical Society. 17, 18

[37] Hao Wang. Proving theorems by pattern recognition I. *Commun. ACM*, 3:220–234, April 1960. 17

[38] C. Mao, T. H. Labean, J. H. Relf, and N. C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, 407(6803):493–496, September 2000. 18

[39] Paul W. K. Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic Self-Assembly of DNA Sierpinski Triangles. *PLoS Biol*, 2(12):e424+, December 2004. 18

[40] R. D. Barish, P. W. Rothemund, and E. Winfree. Two computational primitives for algorithmic self-assembly: copying and counting. *Nano Lett*, 5(12):2586–2592, December 2005. 18

[41] Erik Winfree and Renat Bekbolatov. Proofreading Tile Sets: Error Correction for Algorithmic Self-Assembly. In Junghuei Chen and John Reif, editors, *DNA Computing*, 2943 of *Lecture Notes in Computer Science*, pages 1980–1981. Springer Berlin / Heidelberg, 2004. 18

[42] John H. Reif, Sudheer Sahu, and Peng Yin. Compact Error-Resilient Computational DNA Tiling Assemblies. pages 293–307. 2005. 18

[43] T. H. LaBean, E. Winfree, and J. H. Reif. Experimental Progress in Computation by Self-Assembly of DNA Tilings. In E. Winfree and K. D. Gifford, editors, *5th DIMACS Workshop on DNA Based Computers*, 54 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 123–140, 1999. 18

[44] Hao Yan, Thomas H. Labean, Liping Feng, and John H. Reif. Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices. *Proceedings of the National Academy of Sciences*, 100(14):8103–8108, July 2003. 18

[45] Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. 18

[46] Kyle Lund, Anthony J. Manzo, Nadine Dabby, Nicole Michelotti, Alexander Johnson-Buck, Jeanette Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. **Molecular robots guided by prescriptive landscapes**. *Nature*, 465(7295):206–210, May 2010. 23

[47] Keiichiro Takahashi, Satsuki Yaegashi, Atsushi Kameda, and Masami Hagiya. **Chain Reaction Systems Based on Loop Dissociation of DNA**. In Alessandra Carbone and Niles A. Pierce, editors, *DNA Computing*, 3892, chapter 27, pages 347–358. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. 26

[48] B. M. Frezza, S. L. Cockroft, and M. R. Ghadiri. **Modular Multi-Level Circuits from Immobilized DNA-Based Logic Gates**. *J. Am. Chem. Soc.*, 129(48):14875–14879, December 2007. 26

[49] Peng Yin, Harry M. T. Choi, Colby R. Calvert, and Niles A. Pierce. **Programming biomolecular self-assembly pathways**. *Nature*, 451(7176):318–322, January 2008. 27, 34

[50] Tosan Omabegho, Ruojie Sha, and Nadrian C. Seeman. **A Bipedal DNA Brownian Motor with Coordinated Legs**. *Science*, 324(5923):67–71, April 2009. 27, 34

[51] S. J. Green, J. Bath, and A. J. Turberfield. **Coordinated Chemomechanical Cycles: A Mechanism for Autonomous Molecular Motion**. *Physical Review Letters*, 101(23), 2008. 27, 34

[52] Hongzhou Gu, Jie Chao, Shou-Jun Xiao, and Nadrian C. Seeman. **A proximity-based programmable DNA nanoscale assembly line**. *Nature*, 465(7295):202–205, May 2010. 27

[53] J. W. Lloyd. ***Foundations of logic programming; (2nd extended ed.)***. Springer-Verlag New York, Inc., New York, NY, USA, 1987. 27

[54] Leon Shapiro and Ehud Y. Sterling. ***The Art of PROLOG: Advanced Programming Techniques***. The MIT Press, April 1994. 27

[55] Piotr Wasiewicz, Artur Malinowski, Robert Nowak, Jan J. Mulawka, Piotr Borsuk, Piotr Weglenski, and Andrzej Plucienniczak. **DNA**

computing: implementation of data flow logical operations. *Future Generation Computer Systems*, 17(4):361–378, 2001. 29

[56] Piotr Wasiewicz, Tomasz Janczak, Jan J. Mulawka, and Andrzej Plucienniczak. **The Inference Based on Molecular Computing.** *Cybernetics and Systems*, 31(3):283–315, 2000. 29

[57] John A. Rose, Ken Komiya, Satsuki Yaegashi, and Masami Hagiya. **Displacement Whiplash PCR: Optimized Architecture and Experimental Validation.** In *DNA Computing*, 4287, pages 393–403, 2006. 29

[58] Satoshi Kobayashi. **Horn Clause Computation with DNA Molecules.** *Journal of Combinatorial Optimization*, 3(2):277–299, Jul 1999. 29

[59] Tom Ran, Shai Kaplan, and Ehud Shapiro. **Molecular implementation of simple logic programs.** *Nature Nanotechnology*, 4(10):642–648, August 2009. 29, 30, 34, 55, 61

[60] A. Fire, S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver, and C. C. Mello. **Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans.** *Nature*, 391(6669):806–811, February 1998. 29

[61] Zhen Xie, Liliana Wroblewska, Laura Prochazka, Ron Weiss, and Yaakov Benenson. **Multi-Input RNAi-Based Logic Circuit for Identification of Specific Cancer Cells.** *Science*, 333(6047):1307–1311, 2011. 31, 62

[62] Iñaki Sainz de Murieta and Alfonso Rodríguez-Patón. **Smart Bidirectional Sensors Made of Nucleic Acid Molecules.** In *International Conference on Synthetic Biology*, 1, pages 73–74. GenopoleÂ, 2010. 35, 63

[63] Kensaku Sakamoto, Hidetaka Gouzu, Ken Komiya, Daisuke Kiga, Shigeyuki Yokoyama, Takashi Yokomori, and Masami Hagiya. **Molecular Computation by DNA Hairpin Formation.** *Science*, 288(5469):1223–1226, May 2000. 55, 61

[64] Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki. **Scalable, Time-Responsive, Digital, Energy-Efficient Molecular Circuits**

Using DNA Strand Displacement. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA Computing and Molecular Programming*, 6518 of *Lecture Notes in Computer Science*, pages 25–36. Springer Berlin / Heidelberg, 2011. 56, 61

[65] John H. Reif. The Emergence of the Discipline of Biomolecular Computation in the US. *Biomolecular Computing, New Generation Computing*, 20:2002, 2002. 58